DOCTOR OF SCIENCES

Enhancing Web 2.0 Usability: Handling the Local Contexts of Web Users

Al-Jabari, Mohanad

Award date:
2011

Awarding institution:
University of Namur

FACULTES UNIVERITAIRES NOTRE-DAME DE LA PAIX
FUNDP NAMUR

FACULTÉ D'INFORMATIQUE

Enhancing Web 2.0 Usability: Handling The Local Contexts Of Web Users

Mohanad O. JAABARI

University of Namur - Faculty of Computer Science
Rue Grandgagnage, 21 • B-5000 Namur (Belgium)

Jury: Prof. Jean-Marie Jacquet - University of Namur (President)
Prof. Jean-Luc Hainaut - University of Namur (Promoter)
Prof. Phillippe Thiran - University of Namur (Co-Promoter)
Prof. Michael Mrissa - LIRIS, University of Lyon 1, France
Prof. Stephane Faulkner - University of Namur
Prof. Vincent Englebert - University of Namur

September, 2010
Thesis presented in order to obtain a PhD degree in Science, Computer Science Option

Acknowledgements

I would like to express my appreciation to many colleagues and friends who contributed to this thesis in any way.

First, I would like to thank my thesis advisor Prof. Phillippe Thiran for giving me the opportunity to carry out the research described in this thesis. He motivated me to work on my PhD thesis and also on several research papers reporting on the research results. Working with him has always been a very instructive and a pleasant experience for me. During the week and even in the weekend, and in spite of his overbooked agenda, he was always willing to supervise my work and give me advice. Without his supervision, I would never have been able to achieve this result. He is really an excellent supervisor.

Second, I Would like to thank my co-advisor Prof. Michael Mrissa. . .

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In less than two decades, the World Wide Web has grown into one of the most popular channels that provide information about most aspects of the life (e.g., economic, education, political, etc.). For example, Web users from around the world can *browse* the Web for reading news, purchasing products, registering online courses, forecasting weather, etc. (referred to them as *Web Readers*).

Recently, the emergence of Web 2.0 has revolutionized the way information is designed and accessed over the Web. Web 2.0 sites enable Web users not only browsing the Web but also *creating and updating* Web contents. Hence, they can act as active *Web authors*. In addition, Web 2.0 sites/services can *aggregate* Web contents from several sites and display them together in a single Web page.

However, Web users originate from different communities and they implicitly follow their own semantics (refereed to as *local contexts*) to *represent and interpret* Web contents. As a consequence, the same real world concepts might be represented and interpreted in different ways by different *Web authors and readers*. Such concepts are referred to as Context-Sensitive Contents, or *CSC* for short. For example, the concept of *price* could be represented using different currencies (e.g., Euro, US Dollar) and according to different price formats. Also, *date and time* concepts could be represented using different time zones and according to different formats. This situation leads to several *discrepancies* which Web readers encounter on the Web, as they (need to) follow their contexts to interpret these *CSCs*.

One possible solution is to annotate *CSCs* with semantic metadata (i.e., authors' contexts), so that it becomes feasible for Web browsers to adapt the former to different users' contexts. However, the annotation of Web contents is a complex process due to the lack of annotation means that assist authors to perform this process.

From users' perspectives, the discrepancies that could rise from the interpretation of Web contents and the complexity of annotation process are considered as *Web 2.0*

usability problems. This thesis aims at addressing these usability problems.

The rest of this chapter is structured as follows. Section 1.1 initially introduces the terms *usability and usability engineering* as a framework for our approach. Next, Section 1.2 defines the terms *Web 2.0, local context, and Web 2.0 usability.* Also, an example in developed in this section in order to illustrate and analyze the Web 2.0 usability problems that we intend to address. Afterwards, a brief overview of the techniques used to improve and evaluate Web 2.0 usability are presented in Section 1.3 and Section 1.4, respectively. Section 1.5 discusses the scope of this thesis. Finally, Section1.6 presents the topic of the thesis and a number of related research questions.

1.1 Web Usability: Definition and Engineering Discipline

Several approaches from different computer domains have proposed definitions and categorizations of usability. These are varied according to the models they are based on. Although there is no agreed definition, the term *usability* is acknowledged as a quality factor that are used to assess and/or improve the interactions between users and a computer system.

The International Organization for Standardization (ISO) provides a theoretical usability definition in ISO 9241-110:2006[1] standard. The standard defines usability as "the extent to which a product (e.g., software or Web site) can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction", whereas these criteria refer to the following:

- *Effectiveness:* the extent users perform specified tasks *accurately and completely*
- *Efficiency:* the *efforts* expended (e.g., time) to perform specified tasks effectively
- *Satisfaction:* it is more oriented to subjective attitudes of users and it concerns their *comfort or frustration* when they perform specified tasks.

In Human Computer Interaction (HCI) domain, usability has been viewed from two perspectives. The first perspective views usability as a number of quality criteria which specify the desired level of effectiveness, efficiency, and satisfaction. Then, these criteria are utilized to judge and evaluate the actual result of users' interactions with the user interface of a computer software. Several HCI researches describe these criteria in more details. For example, Nielsen [85] provides five usability criteria: *learnability, efficiency,*

[1]More information available on `http://www.iso.org/`

memorability, Errors/safety, and satisfaction. In addition, these criteria are further decomposed into finer-grain criteria, and then different evaluation methods are used to evaluate them against the actual interaction of users [85, 102, 104].

The second perspective views usability as a number of usability design principles, heuristics, and rules. These are kind of general guidelines that should be followed in order to design a usable software and/or improve the usability of existing one. The most common heuristics are introduced by Nielsen [85]. He proposes ten "usability hurestics[2]" which are considered the ten golden heuristic principles that lead to positive effects on the usability of a software. Also, the ISO 9241-10:1996 introduces another heuristic list called *dialogue principles between humans and information systems.* As its name indicates, this list deals with several users' interaction aspects such as the suitability of interaction for the task, for learning, and for individualization. Also, it deals with the conformity of interaction with user expectations, interactions' controllability, and its error tolerance.

In addition, the term *usability engineering* has been used to achieve the usability goals. Indeed, usability engineering is defined as a discipline that provides structured methods for assessing and/or improving the interactions between users and a computer software (i.e., usability goals) [12, 46, 114]. In general, usability engineering consists of the following three phases:

- *Usability analysis.* In this phase, an external knowledge that affects on the usability of the software being developed and/or evaluated is studied and analyzed. This includes identifying the characteristics of users such as their knowledge and skills. Also, it includes descriptions of tasks to be performed by users in terms of their overall goals and the ways (e.g., steps and actions) users interact with the software to achieve these goals. In addition, the context in which the software is used could be analyzed such as the platform used, hardware, and the physical and social environments.

  After that, the desired level of usability has to be identified as a number of qualitative usability criteria, and each criterion is refined into a number of sub-criteria that are applicable to quantify (e.g., performance speed, errors, etc.). Based on usability criteria, a number of usability problems and the requirements to address them are finally specified.

- *Usability design.* Based on the result of the analysis phase, a number of usability

[2]Usability heuristics are summarized on `http://www.useit.com/papers/heuristic/heuristic_list.html`

improvement means are designed and implemented. This could include applying a number of design principles and usability heuristics on the user interface of the software being developed and/or evaluated. Also, they could include other techniques such as adaptation and annotation. For example, adaptation is a means that could be used to improve users' satisfaction.

- *Usability evaluation.* In this phase, one or more usability evaluation methods are used to evaluate the the actual results against the desired level of usability (i.e., usability criteria).

Usability engineering phases could be performed during software design and implementation or it could be performed after software deployment and use. Also, they could be performed many times to reach the desired level of usability. Indeed, many usability researchers and experts deem appropriate to apply usability engineering phases during design phases in order to avoid expensive and complex re-design, and also after system deployment to ensure the desired usability level while users actually use software (See Section 2.4).

The term *Web usability* has been used to apply usability engineering discipline in the Web domain. In this sense, several Web approaches have refined the definition of usability and usability engineering phases to handle the specific characteristics of the Web. For example, [79] introduces Web usability principles and evaluation methods. Web usability principles refine the usability principles and heuristics that were introduced in [85] in order to guide the design process of Web application. Web evaluation methods provide benchmarks for Web usability verification during and after Web application development. In addition, [46] introduces a Web usability engineering methodology for designing and evaluating adaptive Web-based systems. *To conclude*, Web usability has been used to *evaluate and improve* the Web user interface (i.e., Web pages) and the users' interactions with Web pages to achieve specified goals.

In this work, we focus on users' interactions with Web 2.0 sites from local context perspective. Hence, the term *Web 2.0 usability* is defined as quality criteria that are used to *improve and evaluate* the interactions of users (i.e., authors and readers) with Web 2.0 sites. In addition, the usability engineering phases are applied as a framework to analyse the problems and to achieve the goals of the Web 2.0 usability.

1.2 Web 2.0 Usability Analysis

Usability analysis is the first phase of the usability engineering discipline. In this phase, a number of users' external knowledge, the descriptions of tasks to be performed, and the context of interactions are studied and analyzed, as mentioned above. This section aims at studying and analyzing these aspects during Web users/Web 2.0 sites interactions. It initially introduces the tasks that Web users perform on Web 2.0 sites and defines the characteristics of these users (i.e., local context). Next, the effects of users' local contexts on these tasks are analyzed. Finally, the term Web 2.0 usability is refined into a number of sub-criteria, and accordingly a number of Web 2.0 usability problems are defined.

1.2.1 Web 2.0 and the local Contexts of Web Users

The term *Web 2.0* was officially coined by Tim O'Reilly in [89] as a set of design principles and exemplified by sites such as *Wikipedia, MySpace, Upcoming*. However, several researchers including Tim O'Reilly himself argue that there is no clear-cut definition of this term [10, 37].

The heart idea of Web 2.0 lies into the sharing of Web contents from different sources (i.e., users and sites). Community collaborations and contents mashups are the most common Web 2.0 features [10]. To better understand these features, let us distinguish them from the classical Web (known as "Web 1.0") as follows[3]:

- *Community collaboration.* In Web 1.0, a few *Web authors* create and update Web contents for relatively passive *Web readers*. However, Web 2.0 sites enable Web users not only browsing the Web but also *creating and updating* Web contents in usually self-organizing manner. Updating contents could be performed not only by the original author but also by other authors (e.g., *wiki*).

- *Contents mashups.* In Web 1.0, Web contents on a single Web page are usually belong to one Web site. In Web 2.0, contents from several sites can be *aggregated* and displayed together in a single Web page. Contents aggregation could be performed by users' applications (e.g., RSS browser's plug-in) or by a specific Web site aggregator (e.g., blogs' aggregations on *Technorati* site).

The emerging results of community collaboration and contents mashups could not be achieved by individual users and individual Web sites, respectively. Each user gains

[3] A set of Web 2.0 use cases related to these features will be discussed in Section 3.2 in more details.

more from the systems than he puts into it. Also, one Web site can not satisfy all the users' needs. Contents from different sites are to be aggregated and mixed together to satisfy complex users' requests.

In the meanwhile, the Web gathers billions of Web users from all over the world. These users originate from different communities, and follow their local contexts for interacting with the Web. A *local context*[4] refers to the shared knowledge of a community such as a common language and common local conventions such as currency, keyboard configurations, measurement systems, character sets, and notational standards of writing time, dates, durations, physical quantities, prices [17, 109].

As a consequence, several Web contents (i.e., $CSCs$) might be represented and interpreted in different ways by different *Web authors and readers*. This situation leads to several *discrepancies* which Web readers could encounter, as they (need to) follow their contexts to interpret these $CSCs$. With the Web 2.0, Web contents in *a single Web page* can be represented according to *several* authors' contexts, since they are created, updated, and aggregated from different authors and sites, and thus the discrepancies Web readers encounter increase accordingly.

Example

To illustrate the consequences of users' local contexts and Web 2.0 features on $CSCs$ representation and interpretation, we develop an example presented in Figure 1.1. This example considers several authors and readers from different communities. Also, it considers several tasks (i.e., T1-T7) performed on different Web 2.0 pages in sequential manner as follows:

- A British author creates and publishes a length and a date contents on page A (T1). After that, an American author browses the contents of page A (T2), and then updates the date content created by the British author to *07/09/2009* and publishes it again (T3).
- A Canadian author (from the French speaking community) browses the contents of page B and deletes the date content *2009-09-11* (T4). We consider the page B's contents were created by this author. Next, the length and date contents from pages A and B are aggregated to page C (T5).
- An Italian reader browses the date contents that are automatically aggregated, via RSS engine, from pages A and B (T6). Finally, a French reader browses the date

[4]The terms local context and context are interchangeably used and refers to user's local context

and length contents that are aggregated to page C (T7).

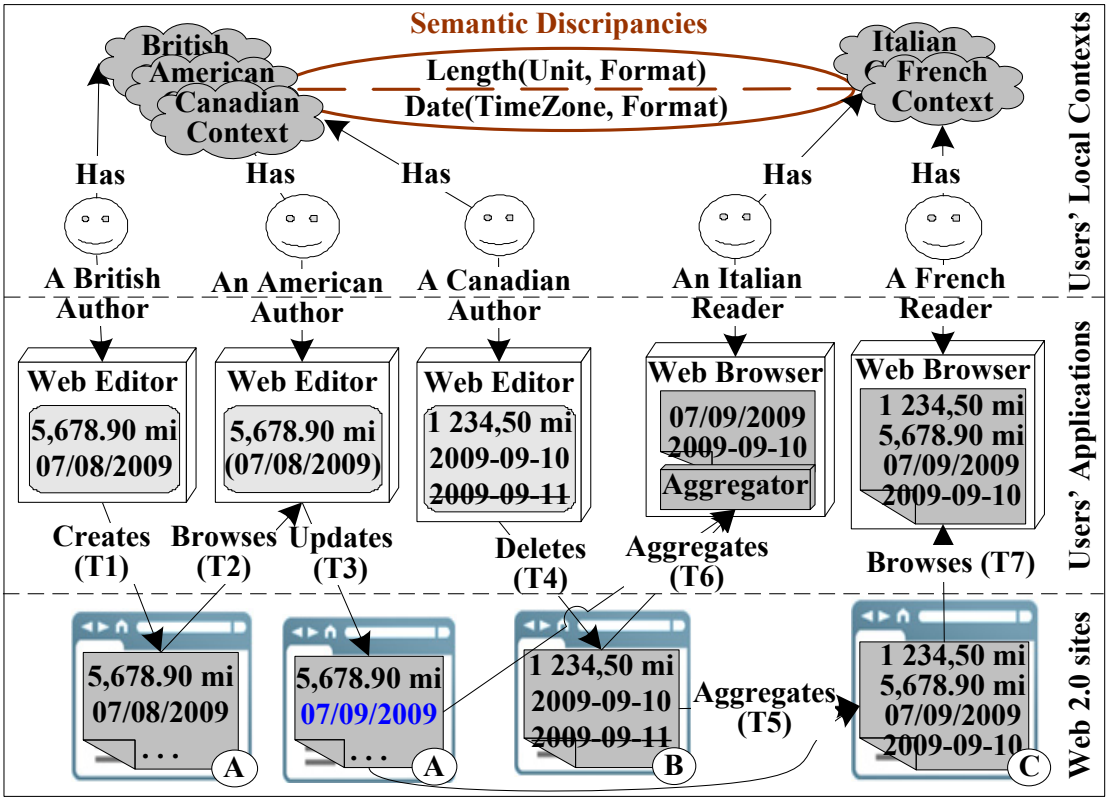

Figure 1.1: Web 2.0 contents' sharing and the implicit usage of users' local contexts

It is obvious that the date and length contents are represented in different ways by different authors. For example, the British author implicitly uses the British context[5] in T1. In contrast, Web readers usually (need to) interpret these contents according to their contexts. For example, as the French reader uses the Meter unit and the French length format (e.g. 1 234,50), he is responsible to adapt the length from Mile to Meter and to French length format.

The problem is similar with respect to the date content 07/09/2009 which is updated at task T3. It is not obvious whether the American author updates the date content according to his context or according to the British context of the original author. Even if the ambiguity is resolved (i.e., he uses his context), the French reader might misinterpret this date as the 7^{th} of September (following the French format) instead of the 9^{th}

[5]Mile unit, British notation (e.g. 1,234.50) and date format (dd/mm/yyyy).

of July (following the American format). Finally, several time zones conventions are implicitly used by different users for representing and interpreting the date contents. These conventions related to the local time used in a specific location on the globe (referred to as time zones). For example, the American author update the date CSC according to the local time of California time zone (i.e., -08:00) and French reader interprets date contents according to the local time of Paris (i.e., + 01:00).

To conclude, the local context is clearly part of the $CSCs'$ semantics. Also, the discrepancies that arise do not relate to the $CSCs$ themselves, but rather to the contexts of Web users who represent and interpret them.

1.2.2 Web 2.0 Usability: Criteria and Problems

As the term Web 2.0 usability refers to quality criteria related to the users' interactions with Web 2.0 sites, the aforementioned issues can be seen as Web 2.0 usability problems. Prior analyzing these issues from usability perspective, the desired usability goals related to users' interactions have to be specified (i.e., usability criteria). To this end, we rely on the usability criteria defined in the ISO 9241-110:2006 standard (i.e., effectiveness, efficiency, and satisfaction) and refine to the quality of both authors' and readers' interactions as follows.

The interactions of Web authors satisfy the effectiveness and efficiency criteria when Web authors be able to represent all Web contents and the local context information (if any) used for representing these contents *accurately and completely*. In addition, the efforts (i.e., time) needed to represent these contents together with the corresponding context information are relatively not too high[6].

The interactions of Web readers satisfy the effectiveness and efficiency criteria when Web readers are able to interpret all Web contents according to their own local contexts. This implies that readers are able to specify their local contents information. Also, they do not misunderstand Web contents created and updated by authors who have different local contexts. Finally, the efforts required to interpret the local contexts of Web authors used to represent Web contents are no longer required, as the latter presented according to their readers' contexts.

The satisfaction criterion is more oriented to subjective attitude of authors and readers. Indeed, this criterion can be seen from two perspectives: work satisfaction and emotional satisfaction. Work satisfaction refers to the ability of a user to successfully achieve a specific task he wants to perform. Instead, emotional satisfaction refers to the

[6]Here we mean if it is compared with the time required to represent Web contents only.

comfort and acceptance of the way a user achieves a specific task he want to perform. Indeed, work satisfaction could not lead to emotional satisfaction [57].

We assume that work satisfaction is included in the aforementioned effectiveness criteria, and therefore the second perspective is more appropriate for this work. In this sense, authors are considered satisfied if they do not face difficulties to specify their local context and to represent Web contents together with corresponding context information (if any) or they reject to do this in the future. On the other hand, readers are considered satisfied if they do not face difficulties to specify their local context and if they interpreting *CSCs* according to their local contexts.

Table 1.1 below refines our understanding of Web 2.0 usability criteria from authors' and readers' perspectives.

Web 2.0 Usability	**Readers Perspectives**	**Authors Perspectives**
Effectiveness	The extent that Web readers can interpret and understand Web contents according to their local contexts accurately and completely	The extent that Web authors represent Web contents and the corresponding local context information accurately and completely
Efficiency	The efforts expended, usually in terms of time, to interpret Web contents effectively	The efforts expended, usually in terms of time, to represent Web contents and local context information effectively
Satisfaction	The readers' comfort and acceptance when they specify their local context and interpret Web contents	The authors' comfort and acceptance when they represent Web contents and their corresponding local context information

Table 1.1: Web 2.0 usability criteria from Readers' and Authors' perspectives.

Based on these criteria and the aforementioned example, the following two Web 2.0 usability problems are defined. First, the interactions of Web authors are *ineffective* when they create and update *CSCs*. Indeed, the representation of *CSCs* are *incomplete*, since part of their semantics (i.e., local context information) are not explicitly represented.

Secondly, the interactions of Web readers are *ineffective and inefficient* when they browse *CSCs*. Indeed, Web readers require additional efforts to interpret *CSCs* that are implicitly represented according to authors' contexts (inefficiency problem) and they might misinterpret these contents (inaccuracy problem), as already shown in the above

example.

Furthermore, the inefficiency and inaccuracy problems would be increased in accumulative manner for two reasons. First, when *CSCs* in a single Web page are created and updated by several authors who have different local contexts. Secondly, when *CSCs* authored from several authors on several Web sites are dynamically aggregated and displayed together in a single Web page. Furthermore, the inaccuracy problem could affect on update and delete tasks. For instance, if a Web author misinterprets Web contents, he could incorrectly update or delete these contents.

Table 1.2 summarizes the Web 2.0 tasks and the Web 2.0 usability problems that are caused by these tasks when Web users implicitly use their local contexts for performing these tasks.

Web 2.0 Tasks	**Web 2.0 Usability Problems**	**Description**
Create	Incompleteness	*CSCs* are implicitly created according to multiple authors' local contexts
Update	Incompleteness	*CSCs* are implicitly updated according to multiple authors' local contexts
	Inaccuracy	An author could incorrectly update *CSCs* created by other authors who have different local contexts
Delete	Inaccuracy	An author could incorrectly delete *CSCs* created by other authors who have different local contexts
Browse	Inefficiency	A reader could incorrectly browse *CSCs* created by one or more authors who have different local contexts
	Inaccuracy	A reader could require additional efforts to interpret *CSCs* represented according to one or more authors' local contexts
Aggregate	Increase inefficiency and Inaccuracy	*CSCs* created and updated according to multiple authors' local contexts are aggregated and presented in a single Web page

Table 1.2: Web 2.0 usability problems and Web 2.0 tasks.

1.3 Web 2.0 Usability Design

Usability design is the second phase of the usability engineering discipline. In this phase, a number of usability improvement means have to be designed and implemented in order to address the usability problems identified in the first phase. To this end, two usability improvement techniques are utilized in this work: Web annotation and Web adaptation. This section defines these techniques and provides a brief overview about them.

1.3.1 Web Annotation

The term *annotation* has been used to refer to the process of adding metadata to documents or contents of documents for explaining their roles, structures, meanings, formats, etc. Also, annotation has been used to refer to metadata itself [111]. Metadata can be attached to different types of documents (e.g., XHTML, PDF, Latex, etc.). Also, metadata can be represented in different languages (e.g., natural, formal, ontology languages, etc.) and with different types of vocabularies (e.g., keywords, taxonomies, etc.).

In semantic Web domain, the term *Web annotation* has been used to annotate Web resources (i.e., anything identified by URI[7]) with semantic metadata, such that a software application can interpret these resources. Semantic metadata are usually defined in ontologies. Also, machine interpretable languages like RDF and OWL are usually used for representing them (See Section 2.3.1).

In this work, we use Web annotation to address the *incompleteness* of Web contents representation. In this sense, *Web annotation* refers to the process of annotating $CSCs$ (e.g., date) with authors' local context information (e.g., date format and time zone) in Web 2.0 domain. Web annotation should include the following tasks to be successfully performed.

- *Identification of target Web content.* The first step is to identify a CSC that needs to be annotated. For example, the date CSC that is created by the British author in the example presented in Figure 1.1.
- *Annotation of the identified Web content.* The second step is to annotate the target Web content with an author's local context information. For example, annotating the date $CSCs$ with the date format and time zone related to the British author's local context. To successfully accomplish this, the relation between each CSC and its corresponding context information has to be specified. In addition, an annotation technology is required to associate them together.

[7] Uniform Resource Identifier

- *Semantic annotation testing.* Since the context information and the target Web contents (or part of them) are intended to be interpreted by users' applications, their representation should be precisely represented (without errors). Otherwise, software application can not interpret them.

From the usability perspective, relying on authors to perform these tasks is complex. Indeed, authors are non experts and therefore they often do not know the relations between *CSCs* and local context information. Also, they do not have theoretical and technical knowledge about semantic metadata and annotation. Furthermore, the result of annotation is intended to be interpreted by software applications. Therefore, authors require extra works to perform these tasks without errors. Also, they could refuse to annotate Web contents in the future.

To sum up, Web annotation address the incompleteness of *CSCs* representation, but it could leads to inefficiency and inaccuracy problems. Table 1.3 refines the Web 2.0 usability problems identified in Table 1.2 by replacing the *incompleteness* problem with the *inefficiency and inaccuracy* problems related to Web annotation.

Web 2.0 Tasks	**Web 2.0 Usability Problems**	**Description**
Create and Annotate	Inefficiency	An author requires extra effort to create and annotate *CSCs* with his suitable local context information
	Inaccuracy	An author could incorrectly represent local context information used to annotate *CSCs* when they use annotation syntax
Update and Annotate	Inefficiency	An author requires extra effort to update and annotate existing *CSCs* with his suitable local context information
	Inaccuracy	An author could incorrectly represent local context information used to annotate *CSCs* when they use annotation syntax

Table 1.3: Web annotation and Web 2.0 usability problems

1.3.2 Interactive Web Annotation

To alleviate the consequences of the inefficiency and inaccuracy problems related to Web annotation, authors have to be assisted to accomplish the tasks of the latter. To this

end, Web annotation is designed to be interactively performed by authors. More specifically, this process hides the relations between $CSCs$ and corresponding local context information. Also, it automates the representation of Web annotation (See Section 5.3).

1.3.3 Web Adaptation

In general, Web adaptation refers to the process of transforming Web page's contents to be suitable to users' contexts. In this sense, Web adaptation can be viewed from different perspectives such as the purpose of adaptation, the types of context information used to perform adaptation, and types of Web contents to be adapted. Also, various approaches from different applications domains utilizes different techniques for conducting Web adaptation process (see Section 2.2).

This work utilizes Web adaptation to address the *incorrectness and inefficiency* of Web contents interpretation. Hence, *Web adaptation* refers to the process of adapting the annotated $CSCs$ according to their readers' local contexts. Web adaptation should include the following tasks to be successfully performed.

- *Web contents parsing.* The first step is to determine which Web contents that need to be adapted (i.e., the context-sensitive contents) and what is the type of these contents (e.g., date content).
- *Local contexts identification.* For each context-sensitive content, the context information of this content and context information of a reader need to identified (e.g., date format and time zone for a date content).
- *Web contents adaptation.* Each context-sensitive content has to be adapted according to its reader's local context in this step.

1.4 Web 2.0 Usability Evaluation

Usability evaluation is the third phase of usability engineering discipline, as aforementioned. This evaluation deals with users' interactions with a software and could be utilized during software design and implementation phase to avoid several usability problems. Also, it could be utilized after software deployment phase to compare the actual users' interactions with the desired usability level (usability criteria). Also, many usability evaluation methods have been proposed and used for conducting these evaluations (see Sections 2.4). This work utilizes the term *Web 2.0 usability evaluation* for the following two purposes:

- *Design inspection of Web annotation and Web adaptation.* Inspecting several usability aspects during the design of Web annotation and Web adaptation. The purpose of this inspection is to optimize the representation and interpretation of $CSCs$ and their corresponding context information according to the ways users can and prefer to do this.

- *User-testing evaluation methodology.* Introducing an evaluation methodology which explains our recommendation on how to evaluate the actual interactions Web 2.0 users after deploying Web annotation and Web adaptation as usability improvement techniques.

1.5 Scope of the Thesis

We mainly focus on the interactions of Web users with Web 2.0 sites from a local context perspective. Indeed, Web authors and readers implicitly use their local contexts for representing and interpreting Web contents, respectively. As already illustrated before, this leads to the following two usability issues. First, the representation of $CSCs$ are *incomplete*. Secondly, Web readers could *inaccurately and/or inefficiently* interpret these contents. Our primary goal is to address these two issues.

We Initially study several types of $CSCs$ and context information which is implicitly used to represent and interpret each type of these content. Our purpose is to identify the relations between these contents and their related context information at conceptual level.

Secondly, we use Web annotation and Web adaptation techniques for handling the aforementioned usability issues. Indeed, Web annotation is used to complete the representation of $CSCs$ by enriching them with semantic metadata (i.e., local context information related to their authors). Also, the annotation process is interactively designed in order to assist non-expert authors to easily annotate $CSCs$. Afterwards, Web adaptation is used to adapt the annotated $CSCs$ from their authors' local contexts to their readers' local contexts. Additionally, an extensive attentions are given to the ways users actually interact with Web 2.0 sites. Our aim is to design the above Web annotation and Web adaptation according to the way these users can and prefer to represent and interpret $CSCs$.

Finally, we introduce an evaluation methodology which details our recommendation on how to evaluate the actual users' interactions after applying the proposed Web annotation and adaptation techniques. In the following, we summarize the basic assumptions

and restrictions underlying this thesis:

- Since our interest is to handle users' local contexts during their interactions with Web 2.0 sites, we focus on Web 2.0 usability problems that hamper the representation and interpretation of $CSCs$. Hence, we do not address other types of users' contexts such as users' preferences and skills or the capabilities of users' devices. Such issues are addressed in several approaches as discussed in Section 2.2.2. Moreover, we do not consider other Web 2.0 usability problems such as problems related to the user interface of Web 2.0 sites or navigation between hyperlinks.

- Web annotation is used to annotate $CSCs$ with context information related to their authors' contexts. We do not consider the relations of these $CSCs$ with other Web contents. Also, the annotation of Web contents (including $CSCs$) with additional information such as information related to their meanings or structures is out of this thesis scope. Several Web annotation approaches have considered these issues (see Section 2.3.2).

- Web adaptation is used to adapt the *presentation* of annotated $CSCs$ according to their readers' local contexts. Hence, we do not aim at providing different Web contents or different navigation links to different users (see Section 2.2.3) .

- Our design relies on the *semantic object* notion given in [82] to identify the relations between $CSCs$ and their related context information. Also, it reuses a number of common ontologies to foster the inter-operability of semantic objects among users' applications (see Section 4.3).

- Web annotation and Web adaptation are designed as extensions to users' applications (i.e., traditional Web editors and Web browsers, respectively). Hence, we do not aim to introduce new users' applications that replace the existing ones. Also, we do not aim at changing the way users interact with Web 2.0 sites. Moreover, Web annotation and Web adaptation are designed to be compatible with existing Web technology stacks. More specifically, we rely on RDFa language to embed semantic metadata inside XHTML document (see Section 5.2). Also, Web adaptation is designed based on the DOM[8] of requested XHTML documents (i.e., Web pages). In this context, we do not aim at using/introducing new technology other than XHTML to represent Web pages' contents. Also, we do not aim at changing the way in which Web pages are requested from and submitted to users. In other

[8]Document Object Model

words, our approach is designed to work on the top of the existing Web architecture and Web technology stacks.

- Finally, conducting the introduced evaluation methodology to practically evaluate the actual users' interactions after applying our annotation and adaptation approach is not addressed in this work (see Chapter 7).

1.6 Thesis Topic and Research Issues

Thesis Topic

We can now define the topic of this thesis as follows:

> *How can we enhance Web usability of Web 2.0 sites by annotating context-sensitive contents with their authors' local contexts and adapting the annotated contents according to their readers' local contexts?*

Additionally, we define four research issues to address this topic. These issues and the contributions related to them are summarized as follows:

Understanding research approaches related to our Work

The best way to understand a problem is to survey approaches and techniques that attempts to address it. In this thesis, we focus on enhancing the usability of Web 2.0 sites using Web annotation and Web adaptation techniques. Hence, *Chapter 2* discusses several works related to the following three subjects: (1) Web adaptation; (2) Web annotation; and (3) Web usability evaluation.

Analyzing the consequences of users' local contexts on their interactions with Web 2.0 sites

This issue is discussed in *Part I* of this thesis. It presents a number of possible Web 2.0 use cases and also a number of Web contents (i.e., $CSCs$) which are represented and interpreted according to the local contexts of their authors and readers, respectively. Next, it analyzes the consequences of this on the interactions of users (i.e., interpretation discrepancies), and the requirements to address these consequences. This part of the thesis has been published in our work presented in [4].

Designing a context-aware approach to handle users' local contexts during their interaction with Web 2.0 sites

This issue is discussed in *Part II* of this thesis. It involves Web annotation and Web adaptation as core components of our approach. Also, it utilizes several design techniques and technologies to support these core components. This work has been published in our works presented in [5, 6, 81].

In *Chapter 4*, we initially evaluate several design alternatives to adapt $CSCs$ according to their readers' local contexts. Then, we present a semantic representation model. This model is based on the notions of semantic objects and local context ontology to enrich (i.e., annotate) $CSCs$ with suitable authors' context information. Finally, we introduce an architecture that illustrates how our approach work seamlessly with Web technology stack.

In *Chapter 5*, we also evaluate several annotation alternatives to annotate $CSCs$. Then, we develop an interactive annotation process. This process assists authors to specify their local context and annotate $CSCs$ with a suitable context information. In addition, an annotation engine prototype is developed as a proof-of-concept corresponding to Web annotation.

In *Chapter 6*, we develop an adaptation process. This process adapts annotated $CSCs$ from their multiple authors' local contexts to their reader's local context. Also, an adaptation engine prototype is developed as a proof-of-concept corresponding to Web adaptation.

In addition, we inspect several usability aspects during the design of Web annotation and Web adaptation. Our intention enable users to represent and interpret $CSCs$ according to the ways they prefer to do this.

Introducing a methodology for evaluating the actual results of users' interactions after applying the proposed design approach

This issue is discussed in *Part III* of this thesis. It introduces an evaluation methodology which details our recommendation on how to evaluate the actual users' interactions after applying the Web annotation and Web adaptation summarized above.

Apart form the above four issues, *Part IV* presents the conclusion of this work and our plan for future work.

Chapter 2

State of the Art

2.1 Introduction

The best way to reach a good understanding of a problem is to survey the research approaches and/or techniques that attempt to address this problem or other similar problems. In Section 1.2.1, we discuss several problems related to the interactions of Web 2.0 users as usability problems. Next, Web annotation and Web adaptation are defined as usability improvement techniques. In addition, Section 1.4 utilizes the term Web 2.0 usability evaluation to inspect several usability aspects during the design of the usability improvement techniques, and also to introduce a methodology for evaluating the interactions of Web 2.0 users after applying these improvement techniques. This chapter aims at discussing several works related to the usability improvement techniques and the Web 2.0 usability evaluation.

The rest of this chapter is structured as follows. Section 2.2 discusses several aspects related to Web adaptation. Section 2.3 discusses Web annotation with more focus on the 'semantic Web' viewpoint. Afterwards, Section 2.4 discusses several approaches and methods that deal with Web usability evaluation. Finally, Section 2.5 situates our approach among other research works.

2.2 Web Adaptation

Several terminologies have been used in several Web domains to describe the goals of Web adaptation. In mobile and ubiquitous application domain, the term *Customization* is used to refer to adaptation of a Web application itself to be suitable to a particular context such as users' locations and users' devices capabilities [68]. In adaptive hypermedia, the term *personalization* has been used to refer to the process of providing different Web contents and/or different navigation links to different users, based on their skills,

preferences, and/or needs. For example, personalization is utilized by several E-learning applications for analyzing student skills and provides suitable learning materials accordingly [32, 94]. Finally, the terms *adaptive and adaptable* have been distinguished based on whether users initiate the adaptation process or it is initiated automatically by an application. The first term implies that adaptation process is initiated automatically, whereas the second term implies that users who initiate the adaptation process [71].

In this work, Web adaptation refers to the process of adapting the annotated $CSCs$ involved in Web pages to be suitable to their readers' local contexts, as already defined in Section 1.3.3. In this section, we aim at discussing research approaches that use Web adaptation. To this purpose, we identify four dimensions (or perspectives) which are related to Web adaptation and discuss Web adaptation approaches accordingly. These dimensions are illustrated in Figure 2.1 and summarized as follows:

- *Web adaptation scenarios.* Based on the purpose of Web adaptation, this dimension identifies three general adaptation scenarios: *personalization scenario, localization scenario, and scenario used in mobile and ubiquitous applications.* Then, it classifies Web adaptation approaches accordingly. This dimension is discussed in Section 2.2.1.

- *Context and context modeling.* This dimension initially discusses several types of context information that Web adaptation based on. Then, it discuss several techniques used to acquire, represent, and store these context information. This dimension is discussed in Section 2.2.2.

- *Types of Web adaptation.* Web adaptation could target Web contents, navigation links, or the presentation of Web contents and navigation links. These types are discussed in Section 2.2.3.

- *Adaptation deployment* considers where Web adaptation is deployed. Here, there are three possible deployment alternatives: client-side, server-side, and proxy-based deployments. These are discussed in Section 2.2.4.

2.2.1 Web Adaptation Scenarios

Personalization Scenario

Personalization scenario focuses on guiding users' navigation and providing personalized Web contents (information and functionalities) based on users' personal information and

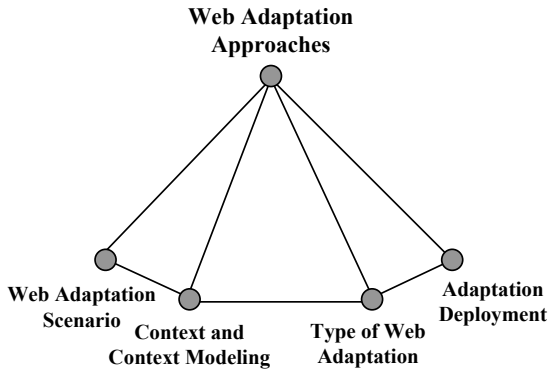

Figure 2.1: Dimentions for Web adaptation approaches

needs such as users' roles, preferences, skills, etc. Personalization scenario is commonly used in E-learning and E-commerce applications.

In E-learning, the goal is to adapt learning materials such as lessons, exercises, and exams based on users behaviors when they interact with E-learning application. For example, [32] proposes an E-learning adaptive model-driven approach for guiding users navigation according to their behaviors (called usage profiles). This approach combines Web Modeling Language (WebML) and UML-Guide. WebML represents the application logics of E-learning web application on server-side, while UML-Guide is a UML state diagram and adaptation predicate rules that acts as personalization engine on client-side. According to UML-Guide state and user profile, Adaptation rules on client-side interact with server-side for requesting relevant learning materials, while preventing irrelevant materials.

Another E-learning recommender approach is proposed in [78]. The goal is to recommend learning contents to learners according to their behaviors (e.g., skills) and navigational activities (also called usage profile). To this purpose, the application uses an ontology for representing learning contents and the relationships between them. In addition, it uses server log file and mining technique for extracting and building learner's usage profile and recommend suitable contents accordingly.

Personalization scenario in E-commerce applications commonly focuses on tracking users' actions, such as recently visited items and past purchase. The goal is to attract and retain customers such as providing personalized offering and support customer interactions. For example, [94] proposes a personalized E-commerce Web application approach. The goal is to provide a personalized navigational view for each user. To do so, this approach uses conceptual, navigational, and user profile models which are

introduced in Object Oriented Hypermedia Design Methodology (OOHDM). Conceptual model represents the application domain objects and user profile model represents user role, user profile history, and his preferences. At run-time, navigational model uses conceptual model and user profile model for providing a personalized navigational view for each user. Another work which provides a comprehensive overview of strategies, benefits, recommendations, challenges, and a set of case studies related to this scenario is presented in [47].

Localization Scenario

Localization scenario aims at adapting Web contents based on specific cultural and local characteristics of users such as local language, local writing format, signs and metaphors. This scenario usually focuses on enhancing the usability of Web pages and considers two main issues: language translation and the roles of users' culture in adaptation.

Language translation is a part of localization, but since it is the most challenges, largest, expensive, and time consuming task; many researches consider it alone. For example, [62] highlights the relationships between language translations and website (a set of web pages) localization. Also, this work develops a strategy that attempts to apply natural language translation approaches on Web site localization.

Roles of users' culture issue focuses on studying the cultural differences between groups of users (called local community), and the role of these differences in localizing web pages. Although the term *culture* can be viewed from different perspectives, [9] defines *culture as the cumulative deposit of knowledge, beliefs, values and attitudes, which rules people's behavior in a society and distinguish the members of one group from another*. In addition, [17] uses the term *culture* to refer to a set of features that distinguish one country or a region of the world from another in the electronic medium of the Web.

The study of local community culture in localization scenario is acknowledged as an important issue in many researches. For instance, [109] argues that community members not only share a common language, but also common culture conventions, such as measurement units, keyboard configurations, character sets and notational standards for writing time, dates, addresses, numbers, currency, etc. In this sense, this work proposes an extension to Web Site Design Method (WSDM) to support the design of localized web site. The WSDM extension specifies the requirements for different community and modeling them as hierarchical locality classes, and use them for providing different Web contents for different communities' members.

Other research fields investigate the impacts of culture differences in localization

scenario [39, 9, 105]. These types of research are commonly based on one of cultural models (also called cultural theory). Cultural models seek to measure the cultural differences between communities according to numbers of variables or factors. One of the most famous cultural models is the Hofstede's model, which categorizes the cultural differences into five fundamental dimensions: power distance, collectivism-individualism, masculinity-femininity, uncertainty avoidance, and long and short-term avoidance.

Mobile and Ubiquitous Applications Scenario

This scenario focuses on adapting Web contents based on the characteristics and environments of users' devices such as device capabilities, network availability, and physical location. Ubiquitous applications (sometimes called pervissave applications) focuses on providing computing and communication services for users at any time, everywhere, and with any communication media. One kind of ubiquitous applications are called adaptive or context-aware ubiquitous applications, which are commonly used in the area of mobile computing [106, 16]. Initially, this kind of applications focuses on providing information and functionalities to users based on their locations. For example, Cyberguide [2] provides information services to tourists about their current location, so that he/she can find directions, retrieve background information, and leave comments on an interactive map. Subsequently, they have been evolved to include other adaptation aspects that are based on device and network capabilities [97, 68].

Other example of this scenario is given in [113]. It proposes a general architecture of context-aware adaptive web information application. This architecture considers three main components of the application separately: the contents (the data to publish), the presentation (the layout of the pages), and the navigation (the hypertext structure of web site). Based on this architecture, the adaptation process operates on all of these components by selecting the appropriate contents (according to user profile), building an adequate layout (according to client device capabilities), and organizing the hypertext structure of the web pages (according to network capabilities).

2.2.2 Context and Context Modeling

The term *context* has been used in various domains. Broadly speaking, the context refers to a set of factors that represent the environment of an application. In computer domain, an often-cited definition of the context is presented in [41]:

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.*

Context-based adaptation implies that context information have to be specified, acquired, and represented. As it is shown in the aforementioned scenarios, there are many types of context information. Also, various techniques and terminologies are used for acquiring and representing these types of information. For example, personalization scenario usually uses the term *user model and/or usage profile* for acquiring and representing users' behaviors, skills, and/or preferences, while localization scenario usually uses the term *culture model* for acquiring and representing local and cultural information related to users. Mobile and ubiquitous scenario commonly uses the term *context model* for acquiring and representing context information related to user's device characteristics, network capability, and/or user's physical location [2, 106]. Other approaches mix several terminologies together to represent different types of context information. For instance, [31] uses both user profile to represent user information and context model to represent device characteristics together with user location and user activities.

To better understand this, the remaining of this section details the *types of context information* that are commonly used in the above adaptation scenarios. In addition, the types of techniques which are used for *acquiring and representing* context information are then discussed.

Types of Context Information

Several research approaches introduces different taxonomies to classify the types of context information. For example, [71] discusses various types of context information such as user and usage, hardware and software, and physical environment information. Also, [67] proposes five context information categories that aims at covering all possible scenarios of adaptive Web applications: *User and Role, Process and Task, Location, Time, and Hardware Device*. Similarly, [68] surveys various types of context information and their related area of usage: user context (used for characterizing user properties and usage behaviors), device and network context (used for characterizing device and network properties especially in mobile application), Location context (used for describing physical location of the users, mostly in mobile and GPS[1] applications), Time context (describes the actual time and time zone variations).

[1] Global Positioning System

In addition, [39] and [17] define cultural information as another context dimension. Indeed, [39] identifies cultural information (language, Layout and menu, symbols and metaphor, contents and structure, Multimedia, color) and proposes a set of hypotheses to investigate how these information relates to general design and localization of web sites. Also, [17] uses the term "culturability" to emphasize the relationship between culture conventions and usability in WWW design. This work concludes that several factors related to users' cultures such as colors, fonts, icons and metaphors, geography, language, flags, and sounds directly affects on the way a user interacts with a Web site.

Based on the above discussion and the aforementioned adaptation scenarios, the following list classifies context information into three general categories. Also, Figure 2.2 illustrates these categories and relates each one with a corresponding adaptation scenario.

1. **User and usage information**. It is commonly used in personalization scenario to form a *user model* and involves information about users' personal details and needs. More specifically, it could involve the following types of information.

   - *User demographic data.* user's personal details such as name, address, birth date and place, education level, etc. Also, it could include user's role such as student or teacher in E-learning applications and customer or manger in E-commerce applications.
   - *User knowledge and skills.* Refers to user's knowledge about application-domain concepts, the relationships between them, and how user interacts with these concepts. For example, it could represent user's knowledge about complex products or services (e.g., printers, ADSL connection, and ticket reservation), and how (s)he deal with them (e.g.: buy, maintain, request). In addition, it could refer to user's skills related to the interactions with learning materials in E-learning applications.
   - *User interests and preferences.* Refers to categories of products, services or the information that a user is interested in (e.g., cars, travel reservation, and sport news). Also, it could refer to specific properties of these categories (e.g., luxury car, economic travel reservation, and football news).
   - *User special needs.* It involves different kinds of information about particular type of users. For example, it could include information about user disability (e.g., visually impaired). Such information is relevant to applications that focus on providing accessible services to such kind of users.

2. **Local and cultural conventions**. It is usually used in localization scenario to form *culture model*. The latter consists of information related to user's local and Cultural conventions. Particularly, it could include the following types of information.

    - *Language*. Refers to community native langauge with all its related issues such as alphabets, number symbols (numerals), direction of writing, and spelling variants.
    - *Location and time zone*. Refers to local geographical location and its related local time variation. Here, geographical location usually refer to geographical region or city of user, rather than actual (physical) location of user.
    - *Notational writing formats*. Refers to local writing formats of information such as time, dates, numbers, currencies, addresses, etc. Different communities use different writing formats. For example, the numeric date format in the united states involves the month value at the beginning (i.e., mm/dd/yyyy), while it involves the date value at the beging in France (i.e., dd/mm/yyyy), as already mention in Chapter 1.
    - *Social conventions and symbols*. Refers to common knowledge that is shared between community members such as signs and metaphors. Signs could include currency symbol, calendar date/time symbols, measurement units, etc. Metaphors refer to implicit knowledge such as special meaning of some terms or signs (e.g., color, icon).

3. **Device and execution environment**. It is commonly related to mobile and ubiquitous scenario and forms a *context model*. The latter could consist of the following types of information:

    - *Device capabilities*. These characterize hardware and software capabilities of user's device. For instance, operating system, browsers version, plug-in availability, java and java script support refer to software capabilities, whereas hardware capabilities involve processing speed, screen size, and input/output devices.
    - *Network capabilities*. These refer to information that characterize the communication media (e.g., wireless network, analog modem, ADSL) and bandwidth that are used to connect to the Web.

- *Physical location.* Refers to physical location of user's device, whereby the latter is commonly represented by GPS coordinates (i.e., GPS longitude and latitude).
- *Physical conditions.* Refers to surrounding environment of user's device such as the level of noise, temperature, and light.

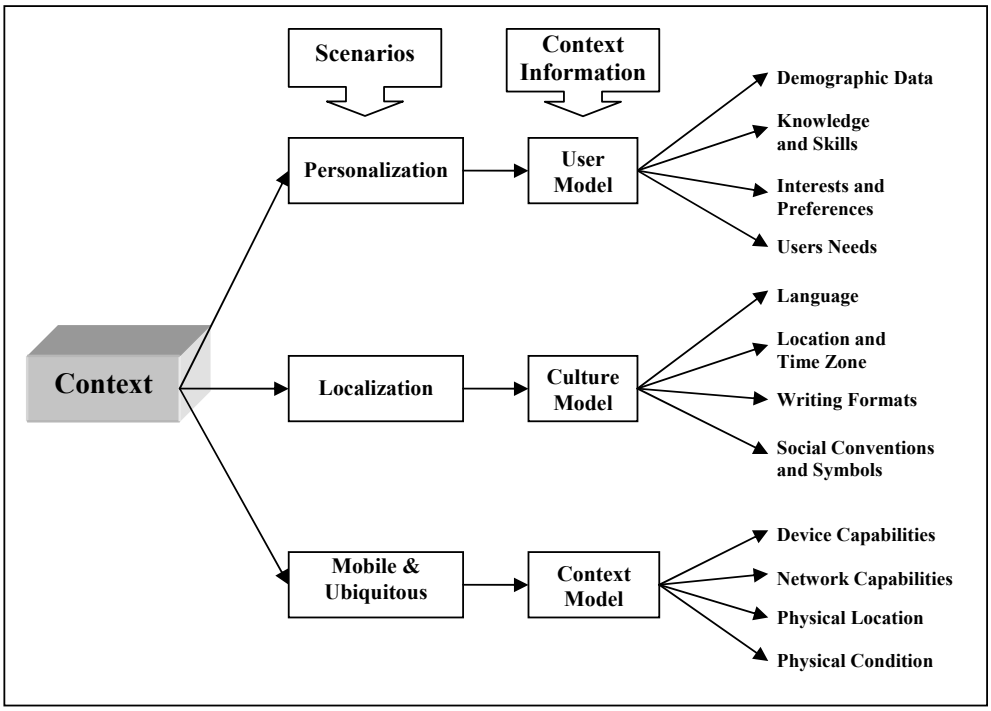

Figure 2.2: Web Adaptation Scenario and Context Information Types

Context Acquisition Techniques

As the types of context information are varied, the techniques used for acquiring them are also varied. For example, user demographic data can be acquired using a Web form interface, and the physical location of user's device can be obtained using GPS sensor. Furthermore, some context information can not be acquired directly and/or can not be used directly to perform adaptation process. For instance, user's behaviors may need to be extracted using extraction or mining technique. Also, it may require processing prior using them to support adaptation (e.g., aggregation) [71]. Therefore, acquisition techniques are classified into several types. In the following, we categories

these techniques into three categories: user driven, application driven, and device or sensor driven acquisition techniques.

1. **User-driven acquisition**

   The simplest context acquisition technique is to let users specifying their context information using a user interface modules such as Web forms. This technique is considered useful for acquiring static information and small amount of context information. For example, many global Web sites such as IBM[2] site use country flag or language abbreviation for providing localized Web contents. Also, My Yahoo Web site uses this methods for enabling users to personalize Yahoo Web contents[3]. However, one of the serious side effect of this method is that users are usually unwilling to provide their personal information or they do not provide correct information due to several reasons such as their privacy concerns.

2. **Application-driven acquisition**

   Since user-driven technique might not be applicable or might be discouraged, several approaches utilize application-driven acquisition techniques. Context information such as user's behaviors are acquired by application, usually based on assumption specified by application provider. In this sense, several application-driven acquisition techniques are used such as acquisition rules, plan recognition, stereotype reasoning, and machine learning (See [71] for more details). *Acquisition rules and stereotype reasoning* are the most frequently used in web adaptation approaches.

   Acquisition rules

   Acquisition rules is a mining technique that observes and acquires user's behaviors based on assumptions pre-specified by application provider. Indeed, acquisition Rules are usually used to collect information about user's dynamic characteristics such as user skills and usage history. This information can not be acquired using user-driven techniques. For example, [32] observes students knowledge and skills via using UML-state diagram, and [78] observes users navigational activities by using server log file and mining techniques.

   Stereotype reasoning

   Stereotype reasoning is an aggregation technique which classifies entities (usually users) into categories based on stereotypes. The latter refers to a set of assump-

[2]available on `http://www.ibm.com/us/`.
[3]available on `http://cm.my.yahoo.com/`.

tions, each of which involves two types of information. First, a set of characteristics that a member should have in order to belong to this stereotype (called member characteristics). Second, a set of Web contents which are associated with a stereotype (called category contents). Here, one category content could be associated with one or more stereotypes, and vice versa.

At run time, Web contents associated with a stereotype are provided to a user as long as this user satisfy member characteristics corresponding to this stereotype. Here, member characteristics refer to context information and it might be aggregated from different type of information such as user's demographic information and user's knowledge and skills. As an example, the work presented in [31] utilizes stereotype reasoning technique. This work aggregates users into groups according to their roles, and integrates each group with one or more site views.

3. **Device or sensor-driven acquisition methods**

   Context information about the environment of user's device is acquired either from user's device directly or from sensors attached with user's device such as GPS sensor and camera. Indeed, user's device can be used for acquiring information related to software capabilities, hardware capabilities, and time zone, whereas sensors can be used for acquiring information related to physical geographical location and physical conditions.

   As a user can use several devices and many users can use the same device, the key issue is how to associate this information with an individual user. Also, some of this information such as hardware capabilities are often difficult to acquire [113].

Web Adaptation Scenarios and Context Acquisition Techniques

Web adaptation approaches usually use more than one acquisition techniques to acquire and/or aggregate context information. This is because context information are varied, as already mentioned. Though, there is a relation between acquisition techniques and Web adaptation scenarios.

In the context of information acquisition, personalization approaches usually combine between user-driven and mining techniques for acquiring user's information and usage behaviors, respectively. Localization approaches commonly use user-driven technique for acquiring local and culture conventions related to user. Finally, Mobile and ubiquitous approaches mostly use device or sensor-driven methods for acquiring context information related to user's device and its environment.

In the context of information aggregation, personalization approaches mostly utilize aggregation techniques for classifying users into categories (usually stereotype reasoning). Also, localization approaches may use aggregation methods for aggregating users into different locality classes[4]. Finally, Mobile and ubiquitous application approaches also use aggregation techniques for aggregating context information from different sources (i.e., different sensors).

Context Representation Techniques

Context representation techniques (also called context modeling techniques) are used for representing and storing context information in machine processable form. As aforementioned, there are many types of context information, each of which has its own characteristics and could be acquired using different techniques. As a consequence, several context representation techniques are also exist.

Context representation techniques can be seen from different perspectives. For example, [106] classifies context representation techniques according to their data structures into six categories: key-value modeling, markup scheme modeling, graphical modeling, object oriented modeling, logic based modeling, and ontology-based modeling. These techniques are then compared with more focus on ubiquitous application needs. The authors conclude that *ontology based modeling* is the most promising approach, since it meets the specified requirements that ubiquitous applications require at best: (1) distributed composition; (2) partial validation; (3) richness and quality of information; (4) incompleteness and ambiguity; (5) level of formality; and (6) applicability to existing environments.

Another approach classifies context representation techniques into general-purpose and project-specific techniques [67]. *General-purpose techniques* usually utilize ontology for representing context information. For example, [107] proposes Context Ontology Language (CoOL). This language describes contextual facts and contextual relationships using aspect-scale-context data model. An aspect identifies a set of scales as a discrete or contiguous context values, and scale refers to a set of objects defining the range of valid context information. Valid context information with respect to an aspect is a value taken from the scale (i.e., range) of this aspect.

Project-specific techniques are used within a specific Web development methodology (Also called Web engineering) such as Web Modeling Language (WebML) [31], Object-Oriented Hypermedia Design Method (OOHDM) [99, 94], and Web Site Design Method

[4]See localization scenario for locality class.

(WSDM) [109]. For Example, context information is represented in OOHDM at two modeling levels: conceptual model and navigational model. *Conceptual model* explicitly represents users' data, roles, preferences, and their relations with application entities. Also, this model specifies a set of constraints (rules) for each user or group of users. Then, *navigational model* (via navigational context model) identifies a set of nodes (page, fragment, and paragraph), links, context classes, and other (nested) navigational contexts that are contained in such context. Based on navigational context model, the application identifies which user or user groups are allowed to view which nodes, links, and context classes in a particular context.

To conclude, context representation techniques depend on several factors such application domain, Web development methodology, and types of context information. Also, finding a flexible and useable context representation technique that covers all cases in one scenario (e.g., personalization) is a challenging task. Context modelers need to evaluate and find out which context representation technique is most relevant to their situations[29, 31, 106]. However, a context modeler needs to take the following criteria (at minimum) when he selects context representation technique:

- *Domain applicability.* Context model should be flexible enough in which it can be implemented in the existing application infrastructure. Also, it should not add extra constraints on application domain. For example, assume user preferences needs to be represented as an event-condition-action rules, whereas an event describes what and action needs to be taken when an action happens. This kind of rules cannot be easily represented in WebML extension[31] or OOHDM [99]. However, it can be easily represented in CoOL [107]. Furthermore, the relations between context model and application models should be precisely and clearly identified. In other word, it should describe which entities from application models are relevant to which entities from context model.

- *Inter-operability.* As long as context information need to be shared between heterogeneous applications, or derived from heterogonous sources; context model should facilitate context sharing and applications inter-operability. In this case, ontology based models (e.g.: CoOL) is best alternative, since context information are formally identified at information level. However, object-oriented model is considered not suitable and do not support applications inter-operability, since context information and corresponding relations are identified at class (object) level.

- *Extensibility* refers to the extent context model can identify and add new context information, without changing the existing context model.

- *Context model decoupling* refers to the extent context model can be represented independently from application logic. For example, Context model in WebML extension (See [31]) is strongly coupled with application logic via set of context classes, while CoOL represents context model independently from application logic as Aspect-Scale facts.

- *Reasoning capability* refers to the extent additional information can be inferred from context model, either from implicit or explicit relations between context information.

2.2.3 Types of Web Adaptation

Web adaptation can be distinguished according to the types of Web contents to be adapted. In this sense, [26] proposes a taxonomy which classifies adaptation processes into content level adaptation and link level adaptation. Also, *Content level adaptation* is further classified into text, multimedia, and modality adaptations. Similarly, *link level adaptation* is further classified into direct guidance, link sorting, link hiding, link annotation, link generation, and map adaptations. furthermore, some of these subcategories are also classified into additional categories. For example, *text adaptation* is classified into natural language and canned text adaptation. Also, *canned text* is classified into inserting/removing fragment, altering fragment, stretch text, sorting fragment, and dimming fragment.

In addition, [67] and [117] extend the above taxonomy in order to involve services provided by Web applications. [67] advocates that adaptation process could address Web services and their parameters, in addition to contents adaptation. For example, the parameters of a flight booking service such the departure and arrival times could be computed according to local time zones of departure and arrival regions.Similarly, [117] extends the above taxonomy with new category called function. *Function adaptation* represents a set of adaptation process associated with one or more services provided this application such as proactive tips, tour planning, and adaptive maps services. For example, spatial Queries is a proactive spatial-context agent that presents tips to a user based on this user's location, nearby objects which user interested in, and user's current interest.

Based on the above discussion, we can specify Web adaptation based on Web contents to be adapted into the following types:

1. **Adaptation of content** refers to adaptation type which provides different Web

contents or services to different users, based on specific type(s) of context information. More specifically, this adaptation could be provided in one of the following forms:

- *Add new information.* New information could be added as extra explanation, additional details, or recommendations. *Extra explanations* are usually used for enriching user knowledge and skills about how to use certain product or consume certain information or services (e.g., printer installation steps or how to buy via MasterCard). Also, it may include small popup explanation that enhances user experience. For example, Google map provides popup image that presents a picture of certain location on the map. *Additional details* give extra information for a user about an item or a product who is interested in (e.g., computer or car characteristics). Finally, *recommendations* inform a user about available offering(s) that he/she may be interested in such as new book release.
- *Delete/hide existing information.* Existing information could be deleted or hidden when it is outdated, user is not interested in it, or when user is prevented to see it. An information such as extra explanation is outdated when a user already consumed it. Also, a user could select part of Web page information as uninteresting information. For example, my yahoo[5] enables users to select information that they are interested in, while hide uninteresting information. Finally, user may be prevented to see some information according to some conditions. For example, some E-learning applications prevent students to access course advanced materials when they lacks knowledge related to introductory materials [32].
- *Alter existing information.* Existing information could be changed according to user location, language, time zone, etc.
- *Alter operational features* change services parameters or services outputs according to user's context. For example, Air-France Web site[6] computes ticket fee and departure time based on currency and time zone of the source country, respectively. Also, it computes the arrival time based on local time zone of the destination country.

2. **Adaptation of navigation** addresses the navigation structures (links) of a Web

[5]http://my.yahoo.com/
[6]www.airfrance.com

site. Particularly, the following three types of hyperlinks adaptations can be distinguished:

- *Generate new hyperlinks.* In this type, new hyperlinks which are related to Web page's contents are discover and added to this Web page. These links could be from other Web sites or just from the same Web site which the page belongs to. Also this type is usually used in recommendation approaches. For example, Amazon Web site[7]). recommends books according to user's interactions and interests.
- *Show or hide existing hyperlink.* Link hiding refers to remove the visible sign (appearance) of hyperlink, so that it looks like as normal text, while the functionality of hyperlink still exist and user can discover it. Hyperlink hiding is usually used to guide users, whereas application assumes that this page is the most relevant information to user, according to his/her level of knowledge or interest. On the contrary, when the application assumes the current page is not relevant to user, it shows the hyperlink to guide his/her to relevant one.
- *Disable/Enable existing hyperlink.* Hyperlink disabling erases the functionality of hyperlink, whereas the sign (appearance) might be still visible. It is usually used in E-learning applications, whereas the application disables and enables hyperlinks of materials according to student knowledge.

3. **Adaptation of presentation**. Adaptation of presentation addresses the layouts and formats of Web contents and hyperlinks without changing their semantics. In other word, Adaptation of presentation represents the same contents and hyperlinks in different way, according to specific context information. For example, changing image to text or reduce image size.

   Notice that, Adaptation of presentation may adapt the user interface of a Web application according to user personal details or needs (personalization), user local characteristics (localization), or according to user device characteristics (mobile scenario).

To conclude, adaptation of contents provides different information and services to different users. In constrat, adaptation of navigation provides different navigation structure and/or changes the functionality of hyperlinks. Finally, adaptation of presentation provides the same Web contents, services, and hyperlinks to different users, but in different formats and layouts. In other words, adaptation of contents and navigations may

[7] http://www.amazon.com/

change the semantic of adapted Web contents and/or hyperlinks, while adaptation of presentation preserves the semantic of adapted Web contents and hyperlinks.

2.2.4 Web Adaptation Deployment

Web adaptation could be deployed on server-side or client-side. Also, it could be deployed on intermediary server called (called proxy server). In the following, we discuss these deployment approaches with more focus on their strengthes and weaknesses.

Server-Side Deployment

Web adaptation is often deployed on server-side. It has been a natural choice for most adaptation processes, since the capabilities on server-side (i.e., software and hardware) is traditionally more efficient [33]. Also, it provides more control and flexibility for Web providers, since the adaptation process can be tied with application logic [113, 20, 75].

The main characteristic of server-side deployment lies into the possibility of conducting both pre-generation and post-generation adaptations. *Pre-generation adaptation* refers to the process of selecting Web contents for each user's requests based on his/her context before generating these contents as a Web page. For example, [78] selects E-learning materials from materials ontology based on student usage behaviors, and then generates them as a Web page. *Post-generation adaptation* refers to the process of transforming Web contents into several versions based on user's context after they are generated as a Web page. Transcoding approaches that transform provided Web contents based on user's special needs (e.g., visually impaired needs) are straightforward example on post-adaptation [18, 55]. In addition, server-side deployment can support all the aforementioned scenarios.

However, server-side deployment has several limitations. The main limitation is that it can only adapt Web contents that hosted on the server which is deployed on it [75]. In addition, it raises important privacy concerns, since server-side application needs to gather and store users' information. In practice, a user is usually enforced to create a user account on server-side in order to make use of Web adaptation. Also, personal information (e.g., users' behaviors) could also be collected and sent to the server-side application without user's knowing about this [84, 11]. Finally, Web users can not control when adaptation process will be performed, in case this process is describes as adaptive.

Client-Side Deployment

Recently, several approaches such as transcoding approaches [18, 55, 115] and rich internet applications [98] deploy Web adaptation on client-side. Client-side deployment is ongoing research field. The main advantage lies into it ability to adapt Web contents that are rendered from different Web sites. In addition, Web users can decide when adaptation process will be performed and also they can control on which Web contents. Also, server-side applications are prevented to collect, store, or share users' information, since there is no need to submit these information for making use of adaptation [84].

However, client-side deployment is less flexible and more complex than server-side deployment. The complexity issue related to the semantic heterogeneity of Web contents, since the latter is usually rendered from different Web sites and therefore have heterogenous semantics. This issue is considered a precondition to conduct a successful adaptation. In addition, this deployment type can consider post-generation adaptation, but it is not common sense and too difficult to consider pre-generation adaptation. Finally, client-side application could have a limited capabilities such as CPU speed and memory storage in mobile devices, and hence can not perform adaptation process.

Proxy-Based Deployment

Web adaptation can be deployed on intermediary server called proxy server. Proxy-based deployment could be used for several reasons. The main reason is to avoid the limitation of server-side or client-side deployments or both. For example, [75] proposes an architecture that deploy adaptation process on intermediary module called *Adaption and Negotiation Module proxy (ANM proxy)*. ANM proxy aims at adapting Web contents that rendering from different server-side, so that it can be presented on different users' devices which have different capabilities. ANM proxy avoid the main limitation of server-side deployment, since it can adapt Web contents from many Web sites. Also, it avoids the limitation of client-side, since users' devices could not perform adaptation process due to its limited capabilities.

2.3 Web Annotation

The term *annotation* has been used to refer to the process of adding metadata to documents, or part of them, for explaining their roles, structures, meanings, and formats. Also, annotation has been used to refer to metadata itself [111]. Metadata can be attached to different types of documents (e.g., XHTML, PDF, Latex, etc.). Also, metadata

can be represented in different languages (e.g., natural, formal, and ontology languages), and also using different types of vocabularies such as keywords and taxonomies.

Annotation is an active and ongoing research field. It has been used in different computer domains such as knowledge management (KM), E-learning, and semantic Web. In these domains, several researches advocate that annotation application should satisfy several requirements. For example, [111] identifies seven requirements for annotation applications that are used in KM (e.g., supporting of standard annotation formats, supporting of multiple ontology, supporting of heterogeneous document formats, supporting of automatic annotation, etc.). In addition, [13] identifies a set of requirements for annotation application that is used to annotate learning materials. This work concludes that annotation application must satisfy three requirements: usefulness, shareability, and usability. *Usefulness* refers to the extent the application allows annotators (e.g., teachers) to annotate different topics to be taught. These annotations should be suitable to the objectives of addressee (i.e., students) and the activities that will be taught (e.g., exercises and lessons). *Shareability* refers to the extent the application enables annotators and addressee communicate through annotations. For example, the annotation must be visually presented to the addressee. *Usability* refers the extent the annotation process does not disturb teaching and learning activities. Finally, [44] introduces eight questions (e.g., are annotations only descriptions?) that Web authors should answer before starting annotation process to support semantic Web.

In practice, several annotation applications have been developed to date and have been classified from different perspectives. For example, annotation applications have been classified based on annotation methods into pattern-based, machine learning-based, and multi-strategy applications [92]. In [36], the features for a list of annotation applications have been surveyed with more focus on the purpose of annotation (i.e., document description, terms description, relations between concepts description) and the annotation interfaces which are used (i.e., annotation editor and ontology browser). The most common classification is based on automation support of the annotation process, whereas the annotation applications have been classified into manual, semi-automatic, and automatic [92, 111].

In this section, we aim at discussing annotation applications that annotate Web documents (i.e., Web pages) and use ontologies for representing metadata annotations. The rest of this section is structured as follows. Section 2.3.1 discusses several dimensions that Web annotation approaches can be distinguished based on. Section 2.3.2 discusses Web annotation approaches. Finally, Section 2.3.3 discusses Web annotation languages that support a specific annotation technique (i.e., internal annotation).

2.3.1 Dimensions of Web Annotation

In order to illustrate the state of Web annotation, this section discusses four dimensions that Web annotation approaches can be distinguished based on. These dimensions are illustrated in Figure 2.3 and summarized as follows.

- *Ontology type.* Describes the type of ontology used for representing Web annotation metadata (i.e., lightweight or heavyweight).
- *Annotation technique.* Describes the type of technique used for annotating Web documents/contents with metadata annotations (i.e., external or internal).
- *Automation support.* Describes the level of automation which annotation application supports to perform Web annotation (i.e., manual, semi-automatic, or automatic).
- *User interface.* Describes the type of interface(s) that enable users to perform Web annotation (i.e., template, semantic auto-complete, ontology browser, etc.).

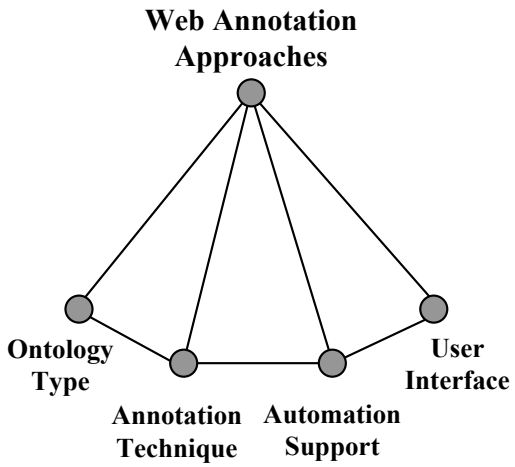

Figure 2.3: Dimentions for Web Annotation Approaches

Ontology Type

Ontology is defined as formal, explicit specification of shared conceptualizations within a specific domain [49, 96]. However, ontologies do not have the same degree of formality. In addition, the components that could be expressed in formal languages such as concept

taxonomies, formal axioms, and disjoint. are not included in all ontologies. In this sense, several researches use the term *lightweight* to refer to less formal ontologies, and the term *heavyweight* to refer to more formal ontologies. In [88], a classification that illustrates different types of ontologies and show the degree of formality is proposed. For example, thesauri or controlled vocabularies are considered as lightweight ontologies (i.e., less formal), while ontologies that have disjoint, inverse, and part-of components are considered as heavyweight ontologies (i.e., formal). Figure 2.4[8] below illustrates this classification (from left to right, the very lightweight, even informal ontologies, to heavyweight ontologies with a large number of formal axioms and constraints).

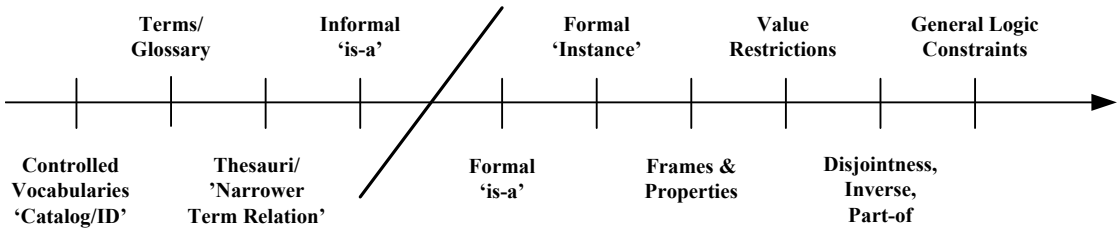

Figure 2.4: Classification of Ontology Types

Ontologies can be represented using different representation languages. Currently, OWL and RDF(S) are the most common languages used. RDF[9] is a language for expressing Web resources (e.g., annotations) based on subject-predicate-object data model. OWL[10] is an ontology langauge based on RDF model that provides more expressive and formal components to represent concept taxonomies, formal axioms, disjoint, etc.

Annotation Technique

There are two main techniques that have been used to conduct Web annotation. These techniques are called external and internal annotation techniques. In external annotation, metadata and Web contents are represented and stored separate documents (referred to as annotation document and annotated documents, respectively). Then, they are associated together using a pointing language such as Xlink and Xpointer. Note that the name *external annotation* comes mainly from the direction of an annotation pointer as it is originated from an external annotation document and refers to a part of the annotated document that represents a Web content to be annotated (i.e., an XHTML tag) [30, 14, 58].

[8]This figure is copied from the cited paper above.
[9]More information available on: `http://www.w3.org/TR/rdf-primer/`
[10]More information available on: `http://www.w3.org/TR/owl-features/`

In contrast, internal annotation stores Web contents and metadata together in the same document. In practice, metadata is usually embedded as an XHTML attribute of the document element (e.g., XHTML tag) which delimits the Web contents to be annotated. To this end, several XHTML language extensions such as SHOE [56], RDFa, and Microformats [3] have been proposed and utilized (see Section 2.3.3).

Both of these techniques have several advantages and limitations. We will discuss these techniques in more details in Section 5.2.

Automation Support

Web annotation could be performed by annotators or by annotation applications, as aforementioned. By automation support, we mean the extents that annotation applications support Web annotation process. To better understand this, let us consider the general tasks which have to performed for annotating a Web content. The first task is to determine the target Web content to be annotated. For instance, a date content. Also, metadata such as date format which is used to annotate the latter is also has to be determined. Next, metadata has to be associated with the target Web content using annotation syntax. Finally,the syntax of annotation needs to be tested and validated, as it is intended to be interpreted by an application.

Based on the above tasks, Web annotation can be classified into three types: manual, semi-automatic, and automatic. *Manual annotation* is the most basic technique. Web annotators are responsible to perform the tasks of annotation process manually. They need to know which Web contents need to be annotated, and what are the corresponding metadata. Furthermore, they are responsible to associate them together using annotation syntax. Finally, they are also responsible to test and validate the correctness of annotation syntax.

Manual annotation faces several significant drawbacks. It is time consuming and prone to errors. Furthermore, Web annotators could be unfamiliar with annotation syntax used (e.g., Xpointer, Microformats, etc.). Therefore, manual annotation is too difficult.

Semi-automatic annotation automates one or more tasks presented above. Here, annotation application usually analyze Web document/contents to be annotated, determine Web contents to be annotated, and suggest/recomend one or more metadata for each content using information retrieval and extraction techniques [112, 61, 28]. After that, an annotator has to confirm the suggested metadata, or reject it. Semi-automatic is less expensive and less errors-prone than manual annotation. However, human interventions

are still required to clarify and test ambiguous annotations.

Automatic Annotation automates all the tasks of annotation process without users interventions. Here, automatic annotation could be used to annotate unstructured or structured Web contents. For unstructured contents, annotation applications usually analyze Web documents to identify the target Web contents to be annotated and automatically annotate them using machine learning and natural language processing (NLP) techniques [53, 34]. For structured contents, annotation applications usually generate a set of templates based on predefined ontological schema. When Web authors fill these templates with Web contents, the application automatically annotate them accordingly [58].

Automatic annotation based on template can't be used to annotate unstructured contents. On the other hand, the quality (i.e., annotation correctness) of automatic annotation that is based on NLP and machine learning is questionable. Indeed, these techniques face the problem of concept disambiguation. One Web content could has several semantics and annotation application could not correctly recognize all of them [34]. Therefore, human interventions are still required to clarify and test ambiguous annotations.

User Interface

Web annotation (i.e., external and internal) could be too complicated with a simple XHTML tag editor. Therefore, most annotation applications provide annotation interfaces that hide the technical complexities of Web annotation process. These interfaces could be simple such as *annotation by selection and ontology browser* or could be supported with more advanced features such as *semantic autocomplete and template generation*. In addition, annotation interfaces could be integrated with Web document editors to support annotation at authoring/editing time or integrated with browsing applications (e.g., Web browser) to support annotation after authoring/editing time (i.e., document editing not possible). Annotation interfaces can be described as follows.

- *Annotation by selection.* It is the most basic interface. Annotators select (highlight) Web contents to be annotated and select corresponding metadata. The latter is usually selected from metadata menu (i.e., tool bar menu or pop up menu) [77, 65].

  Annotation by selection could be useful when an annotation ontology is lightweight and has a small number of metadata. Also, when annotators know which type of Web contents need to be annotated. However, Annotation by selection is time

consuming and very difficult, specifically for non-expert annotators. In addition, it is too difficult to select metadata annotation if the annotation ontology multiple branches with heavy numbers of metadata.

- *Ontology browser.* It is a tree-fashion browser that presents metadata among the hierarchy of ontology. Ontology browser enables annotators to select the appropriate metadata annotations for annotating the target Web contents [103]. Also, it helps annotators to become familiar with ontology metadata, their properties, and their relations with other metadata. Furthermore, it helps annotators to discover more suitable metadata annotation through browsing metadata hierarchy.

  The drawback of ontology browser is still time consuming. Annotators may go through numerous branches to find the appropriate metadata, specially when the ontology in use contains multiple branches with heavy numbers of metadata.

- *Semantic auto-complete. Auto-complete* is an interface feature that allows user to type a few characters, while the application display a set of suggestions that most likely contains the term he/she is looking for. Here the suggested terms are usually based on syntactic matching of the typed characters with terms from a specific data source.

  Semantic auto-complete is used for matching terms with ontological metadata based on the terms' semantics. For example, helping user matches the term *Paris* with one of ontological metadata such as *capital of France, Paris Hilton, name of hotel* [61].

  In annotation applications, semantic auto-complete has been used for two purposes. First, suggesting/recomending metadata for Web content to be annotated when annotator edits or highlights it. Second, helping annotators to find metadata annotation among ontology hierarchy.

  The performance of semantic auto-complete is better than the above two interface. However, it face several drawbacks. First, suggested metadata could not satisfy annotators' requirements. Second, properties of suggested metadata annotations and their relations with other metadata can not be illustrated. Finally, it can not help annotators to discover more suitable metadata annotation among ontology hierarchy.

- *Template-based interface.* it is an interface that is generated based on predefined ontological schemas. The term *template* is used to separate document contents

from documents structure and presentation. In this sense, annotation applications annotate documents' structures and/or presentations when they generate these template[11].

2.3.2 Web Annotation Approaches

A Web document is considered as structure plus contents. Document's structure are constructed from a set of contents' segments and hepertext links. The latter connects Web content or content's segment with other Web content or content's segment [30]. Since Web annotation describes the semantics of Web documents explicitly, it can describe document's structure, contents semantics, or relationships between contents.

Based on the above dimensions and the purpose of Web annotation, this section classifies Web annotation applications into three annotation approaches: *documents annotation, contents annotation, and concepts relations annotation.* Also, it gives some examples on each category.

Documents Annotation

Documents annotation approach describes a set of document properties (e.g., document type, document subject, document structure, etc.) or properties of contents' segments (e.g., segment role) without describing the semantics of Web contents (e.g., meaning). Several Web applications from different domains such as electronic document management (EDM), E-learning, and Web adaptation (i.e., transcoding) utilize this approach.

Saha [112] is an EDM that uses a set of predefined OWL-based annotation schemas to describe the types of Web documents (e.g., Article, form, blog, image, video, etc.) and the structure of each document type. Saha allows author to select the type of document to be authored. After that, it automatically generates the corresponding annotation template based on the annotation schemas. Finally, it generates annotated document when the author fill the annotation template. Web annotation in saha is used as a semantic document index to categorize documents or to search a particular document (e.g., article) among a group of similar documents. SMT[12] [69] is another application similar to Saha. SMT allows users to annotate documents or document's segments from the Web (e.g., report of meeting) by filling an OWL-based predefined metadata template (e.g., meeting template).

[11] See automatic annotation above for more information
[12] Semantic Markup Tool

In E-learning, [14] proposes external-based annotation language and light-weight annotation ontology for annotating learning documents at three levels: pedagogy, domain, and document level. Document annotation approach is used at document level. Teachers can annotate documents presentations (e.g., font color, size, etc.) and documents' segments roles (e.g., title, subtitle, paragraph, etc.) using annotation language and annotation ontology.

SADIe [18] and [14] use documents annotation approach for Web adaptation (i.e., transcoding). SADIe is manual annotation application that uses external annotation technique and annotation by selection interface to annotate documents' segments with simple (i.e., lightweight)transformational metadata (e.g., priority, group, etc.). After that, transcoding engine use these annotations to adapt (restructure) documents' segments in order to make them accessible to visually impaired users. Also, [14] proposes an approach similar to SADIe, but, in addition of transcoding according to users' special needs, this application aims to adapt (transcode) document's segments according to users' device capabilities.

Contents Annotation

Contents annotation approach describes the semantics of Web contents rather than the properties of document or contents' segments. Annotation applications that use this approach mostly aim at automatically identifying (describing) the meaning of Web contents. The latter could be based on natural language knowledge base (e.g., WordNet) or based on specific ontologies (called domain ontologies).

MnM is a manual and (semi)automatic annotation application [36]. It proposes ontology browser as a Web browser plug-in. Ontology browser can load any ontology represented in RDF(S), OWL, and OCML[13]. For manual annotation, Web annotators can annotate contents terms with metadata from the ontology that is loaded on ontology browser. For automatic annotation, MnM uses information extraction (IE) engine, called Amilcare, to detect concept instances appearing in Web documents. This engine must be trained in order to generate annotation rules. Then, the latter is used to automatically extract and annotate similar Web contents from other Web documents.

C-PANKOW is an automatic annotation application [34]. C-PANKOW puts Web content into several linguistic patterns, based on WordNet thesaurus, and automatically annotate it by counting the numbers of occurrence of these lignitic patterns on the Web using google API. XPCOM [43] is another annotation application that uses natural lan-

[13]Operational Conceptual Modeling Language

guage processing (NLP) to identify the meanings of Web contents. Finally, E-commerce applications such as E-travel agencies could use domain ontology such as IATA[14] to annotate airports with their corresponding codes.

Concepts Relations Annotation

Concepts relations annotation approach is usually used to refer Web contents to ontology concepts, identify a set of contents' attributes, and/or identify their relations with other ontology concepts. In this sense, concepts relations approach go a step further than contents approach.

This approach has been initially used in [56]. It proposes manual annotation application that allows authors to annotate the contents of Web pages as an instances of ontology concepts, identify the relations between them, and describe contents' properties (e.g., datatype) based on SHOE. The latter is a set of Simple HTML Ontological Extensions (i.e., internal annotation language).

Ont-O-Mat is also manual application based on CREAM[15] annotation framework [52]. Ont-O-Mat annotates Web contents with heavyweight ontology concepts and define relations between them using Xpointer (i.e., external annotation). For example, the contents *Siegfried Handschuh* and *Steffen Staab* can be annotated with *PhD-Student* and *Lecturer* ontology concepts from SWRC[16], respectively. Also, the links between them can be annotated using *cooperateWith* SWRC relationship. S-CREAM [53] is a semi-automatic extension to Ont-O-Mat that uses IE engine. Initially, annotators need to annotate Web contents manually, so that the IE engine learns the rules of information extraction. After that, this engine can annotate other Web contents automatically.

Recently, many semantic Web 2.0 approaches such as semantic tagging and semantic wikis have used concepts relations annotation. Semantic tagging is a special type of Web annotation that categorizes Web contents into semantic subjects (i.e., ontology concepts) in order to facilitate information search.

SemTag is an automatic semantic tagging application built on seeker platform [42]. It has tagged 264 million Web pages with 434 million semantic tags. SemTag scans Web pages' contents, finds and disambiguates tag instances, and finally annotates them with tags from TAP knowledge base. TAP covers 12 categories (e.g., Authors, Autos, Health, Movies, Places, etc.) with approximately 72,000 tags [51].

[14] `http://www.iata.org/index.htm`
[15] CREAtion of Metadata
[16] Semantic Web for Research Communities, available on: `http://ontoware.org/projects/swrc/`.

Semantic wikis are extensions to traditional wikis that exploit semantic Web technologies (mostly Web annotation), so that software applications (e.g., Web browser) can organize, share, and exchange wiki contents. For example, semantic MediaWiki enables wiki authors to annotate wiki contents with RDF/OWL metadata annotations (e.g., FOAF, Dubline Core metadata, etc.) using RDFa internal annotation language and a simple WYSIWYG[17] editor [74].

SweetWiki [28] is another semantic wiki that uses WYSIWYG editor together with semantic autocomplete and annotation syntax checker for both contents and metadata annotations editing (i.e.,internal annotation). Metadata annotations could be extracted and exploited from other applications (e.g.,linked data[18].) or OWL-based wiki concepts (e.g., wiki word, wiki page, etc.) could be used.

2.3.3 Internal Web Annotation Languages

As aforementioned, internal annotation technique is used to embed metadata inside XHTML document, so that the latter become machine interpretable. To this end, several languages have been proposed and used. For instance, $(KA)^2$ [22] and SHOE [56] are two internal annotation languages which have been utilized in knowledge management domain. $(KA)^2$ extends HTML language with *ONTO* HTML property, whereas the value of this property refers to ontology concept URI (e.g., <a ONTO =‘concept URI’/>). SHOE is a set of Simple HTML Ontological Extension that is used to describe Web contents as instances of ontology concepts, identify the relations between concepts' instances, and specify contents' properties (e.g., data types). With the emergence of the semantic Web, several technologies have been proposed to support internal annotation such as eRDF, Microformats, RDFa, and Microdata.

All of the above technologies share common features: they propose extensions to XHTML langauge in order to annotate XHTML documents with semantic metadata. In the following, we will focus on Microformats and RDFa, since they are currently the main emerging technologies.

Microformats

Microformats aim at standardizing the semantics of Web contents. Microformats propose a set of standards, or *specifications*, and reuse XHTML attributes (e.g., *id* and *class*) to embed those specifications into XHTML documents [3, 108]. For example,

[17] What You See IS What You Get.
[18] More information available on: `http://en.wikipedia.org/wiki/Linked_Data`

hCard specification identifies vocabularies, based on the vCard[19] standard, that provide semantic information about people and organization.

RDFa

RDFa provides more abstract solution that aims at expressing RDF statements in XHTML documents. More precisely, RDFa provides a collection of XHTML attributes (reuses a set of existing XHTML attributes such as *content* and *rel* and introduces new ones such as *about* and *property*) to embed RDF statements in XHTML, and provides processing rules for extracting RDF statements from XHTML [3, 108]. As aforementioned, RDF statements are used for representing semantic information about Web contents (called Web resource) based on a subject-predicate-object model.

Microformats and RDFa: Common and Distinctive Features

Since Microformats and RDFa languages share the same purpose, they share the same common features. However, each language addresses this purpose in different ways. Thus, they have distinct features.

In this context, [108] discusses five design principles related to semantic metadata: authoritative data, expressivity and extensibility, do not repeat yourself (DRY), data locality, and existing contents fidelity. Also, the author compares Microformats, RDFa and eRDF[20] technologies based on these principles. Another discussion is presented in [48]. This work compares between both Microformats and RDFa with more focus on DRY and visible metadata principles. Finally, [3] presents four principles for embedding semantics into HTML: independence and extensibility, DRY, locality, and self-containment.

Common Features

- **Do not repeat yourself (DRY)**

DRY principle says that every piece of information should have one authoritative, unambiguous representation. DRY principle (also known as a single point of truth) aims for reducing duplication as much as possible, since duplication leads to serious problems such as difficulty of change, ambiguity, and increases the opportunities of inconsistency.

In the context of Web annotation, DRY principle means that metadata annotation should be represented *once*. Also, any changes in metadata annotation and annotated

[19] More information available on: `http://www.imc.org/pdi/vcard-21.txt`.
[20] embedded RDF.

Web content should be occurred *once* for both users and their applications [70, 3, 48]. In this sense, RDFa is fully support Dry principle, while Microformats are partially support this principle.

In *RDFa*, annotated contents are represented to be interpreted by users and users' applications. Indeed, Web users interpret Web contents that are represented by XHTML tags, and users' applications interpret the same Web contents as RDF objects. Therefore, any changes in Web contents will be for both users and their applications.

Microformats represent metadata annotation *once*. However, Microformats sometimes use two versions of Web contents: one for users and one for their applications. Therefore, any changes in one version requires changes in the other version (see Section 4.2).

- **Data locality**

Data locality refers to localize Web contents and their corresponding metadata in the same location, so that users or users' applications can "copy and past" a fragment of Web page's contents without affecting on the semantics of this fragment [3, 108]. In this sense, both technologies support relatively[21] locality principle. For example, users'applications can extract and export Microformats specification or extract and export one or more RDF statements from a Web page.

- **Semantic metadata visibility**

In (X)HTML, the term *Metadata* such as meta tags and Cascade Style Sheet (CSS) are designed to help search engine to rank Web pages, and to describe the presentation of Web page's contents, respectively [7].However, *Semantic metadata* are designed for representing the semantic of Web contents in machine interpretable manner, as aforementioned.

In addition the difference in purposes, semantic metadata are designed for enabling users' applications to extract and present them to Web users as well as exchange them with other software. For example, FireFox Tails Export extension[22] allow users to extract Microformats metadata, and export them to other desktop application (e.g., Address Book). This feature is called semantic metadata feasibility. Both Microformats and RDFa are designed to support this features.

[21] Here, Relatively means that users or users'applications have to take a complete fragment (e.g., paragraph, Weblog, Widget, etc.).

[22] Available on `https://addons.mozilla.org/en-US/firefox/addon/4106`.

This feature provides several benefits: (1) providing additional levels of information for users; (2) assisting authors to keep semantic metadata up-to-date, since hidden metadata could be easily forgotten; (3) semantic metadata visibility makes sure that they are relevant to users as well as machine [3, 48].

Distinctive Features

Semantic interoperability and semantic extensibility are the main two features that distinguish the two technologies. Microformats are fully interoperable, since they standardize the semantics of Web contents that are rendered from different server. However, Microformats are inextensible and do not fulfill all authors' scenarios. This is because Microformats propose a finite set of specifications [81]. Furthermore, Introducing new Microformat specification require extensive discussion with the Microformats community for a general (i.e. worldwide) adoption. Until this point is reached, Microformats parsers on users' applications could not interpret what are considered as "exotic" Microformats specifications.

On the contrary, RDFa is fully extensible. Web authors can create their own semantic metadata and extends others. However, exchanging and interpreting RDFa vocabularies from different Web sites require a prior semantic reconciliation, since each sites represents Web contents and semantic metadata in different ways. Ontology mapping, from local ontologies to a shared ontology (e.g.: Dublin Core[23] and FOAF[24]), is a commonly used way to reach semantic interoperability.

Microformats and RDFa: Side by Side Comparison

Table 2.1 summarizes similarities and differences between Microformats and RDFa languages.

2.4 Web Usability Evaluation

Web 2.0 usability, as defined in Section 1.2.2, refers to the effectiveness, efficiency, and satisfaction of users' interaction with Web pages (i.e., authors' annotation and readers' interpretation of Web contents). Also, usability engineering phases are utilized as a framework to *improve and evaluate* this interaction. Indeed, a number of usability criteria and problems related to users' interaction are identified in Section 1.2. Afterwards,

[23] More information available on `http://dublincore.org/`.
[24] More information available on `http://xmlns.com/foaf/0.1/`.

Feature	RDFa	Microformats (MF)
Purpose	Annotate Web contents with RDF-based metadata	Annotate Web contents with MF-specifications metadata
Purpose achievement	Reuse a set of existing XHTML attributes (e.g., rel) and introduce new ones (e.g., about)	Reuse existing XHTML attributes (e.g., class and id)
(X)HTML compatibility	XHTML+RDFa	HTML 4.01/5, XHTML 1.x/2
Semantic visibility	Yes	Yes
Follow DRY	Yes	Partially
RDF/XML mapping	Yes	Custom mapping for each MF
Namespace	XML and CURIE namespace	Flat namespace
Domain applicability	Yes	Limited to predefined MF
Interoperability	Require semantic reconciliation	Predefined MF is fully interoperable
Extensibility	Yes, reuse RDF data model	New MF requires community acceptance

Table 2.1: Side by side comparison between Microformats and RDFa

Web annotation and Web adaptation are identified in Section 1.3 as improvements means to handle these usability problems. Finally, in Section 1.4, the term Web 2.0 usability evaluation is utilized in order to consider several usability aspects during the design of the improvement means and to explain a methodology for evaluating the actual result of users' interactions after deploying these means.

This section aims at discussing the state of the art related to Web usability evaluation. In this context, several researches have classified usability evaluations from different perspectives. For example, [79] distinguishes between formative and summative evaluation based on the development life cycle stage in which the evaluation is performed. *Formative evaluation* are often performed at design stage and aims at checking the understanding of design team for users' requirements before implementing the final Web site. *Summative evaluation* takes place after deployment phase and aims at detecting the usability problems that users actually encounter. In addition, a wide range of evaluation methods have been proposed and used to conduct these evaluation. For instance, [57] classifies evaluation methods into inspection methods and user testing methods based on who perform the evaluation. In inspection methods, a number of *developers or usabil-*

ity experts conduct the evaluation, whereas a set of *actual users* (called representative users) use Web site to perform specific tasks (called representative tasks) in user testing methods.

In practice, formative evaluation usually utilizes one or more usability inspection methods to detect potential usability problems at design time, while summative evaluation utilizes one or more usability testing methods to detect potential usability problems that representative users actually face them after the deployment of a Web site.

Another classification is introduced in [21] and based on the purpose of evaluation. This classification introduces four types for Web site usability evaluation as follows:

- *Objective performance.* Measures the capability of Web site visitors in terms of time taken to complete specific tasks on the site (i.e., task efficiency).
- *Subjective user preferences.* Assess how much the users like the Web site by eliciting their opinions, or by asking them to rate the site (user's satisfaction).
- *Experimental evaluation.* Based on controlled experiments to test the hypotheses about design and their impact on users' performance and preferences.
- *Direct Observation.* Monitor the users' behaviors while interacting with the Web site to detect usability problem.

Finally, an extensive survey of 132 usability evaluation methods is introduced in [63], 75 methods applied on WIMP [25] and 57 methods applied on Web user interface. This survey proposes a taxonomy with four dimensions. Each dimension describes the usability evaluation methods from a specific perspective as follows:

- *Method class.* Describes the type of evaluation conducted at a high level (i.e., user testing, inspection, inquiry, analytical modeling, and simulation).
- *Method type.* Describes how the evaluation is conducted within a method class. For example, user testing class could use thinking-aloud protocol or Web log file analysis.
- *Automation type.* Distinguishes whether the usability evaluation process are automated or not. Also, Describes to which extent this process are automated.
- *Effort level.* Describes the type of effort required to execute usability method. For example, the automation of usability evaluation or part of it usually requires developing a usability evaluation model.

[25] Windows, Icon, Mouse, Pointer

Based on the above classifications and the source of usability data, we can group Web usability evaluation into two approaches: user-testing evaluation and usability inspection evaluation [57, 85]. The following subsections briefly discuss these two approaches.

2.4.1 User-Testing Evaluation

User-testing evaluation aims at setting up experiment with actual users during Web site use in order to compare the actual result achieved with the desired usability level and therefore detecting potential usability problems (if any). Indeed, user-testing evaluation could be used to evaluate several aspects of users' interactions such as objective performance (i.e., task efficiency and number of encountered errors) and subjective user preferences (i.e., users satisfaction), as aforementioned.

In practice, setting up user-testing evaluation have to be carefully planned and managed. This involves identifying a number of evaluation activities and describe the best ways for performing these activities. In this context, many usability approaches advocate that the following activities are necessary for conducting a successful user-testing evaluation. First, the goal(s) of the evaluation and the desired way to reach them have to be specified. Secondly, a number of representative users and a number of tasks or scenarios to be evaluated have to be identified. Thirdly, the types of data related to usability evaluation have to be specified. These data have to be collected while representative users perform the identified tasks or scenarios. Finally, the collected data have to be analyzed in order to compare the actual result achieved by representative users with the desired usability level [46, 79, 85, 101].

In addition, many methods have been used to conduct user-testing evaluation or part of it. These methods are mostly conducted in usability test labs and also with presence of one or more evaluators. Thinking aloud protocol, controlled experiment, and query techniques are the most common methods. The following discusses the main ideas of these methods.

- *Thinking aloud protocol* (also called scenario-based testing) is an informal method which involves a number of representative users who are asked to perform specific tasks, or predefined scenarios, designed to cover the major functionality of the Web site and to simulate expected real-life usage patterns. For each task or scenario, a number of evaluation measures are also designed to cover the intended useability aspects to be evaluated. Such evaluation measures usually target whether the participants correctly accomplished the tasks, how many errors are performed by users, and how many time taken for accomplishing each task.

In practice, users are observed by an evaluator or they are asked to think out aloud (i.e., verbalizing their thoughts) while performing tasks to be evaluated. Then, the collected data (i.e., actual result performed by representative users) are compared with the desired usability level based on the specified evaluation measures [21].

- *Controlled experiment* is a formal explantation method, whereas one or more hypotheses are proposed (called usability hypotheses) and actual experiments are carried out to accept or reject these hypotheses.

  Controlled experiment is recommended to evaluate the result of Web adaptation approaches such as personalization applications. In this sense, a number of representative users are asked to make a specific task that is based on adaptation twice: one before applying adaptation and one after applying it. Then, the effects of adaptation on users' behaviors are measured [73]. In addition, it might be used to compare the result Web site interface (i.e., Web page) or users' interactions with specific tasks before and after applying usability improvement means.

- *Query techniques* is a method that is mostly used to collect subjective data about users' opinions and satisfactions related to the layouts of Web pages and also related to their interactions with these pages after performing specified tasks. The most common query technique used in Web domain is called Web-based or online questionaire.

  A questionnaire is basically a number of predefined questions with different styles such as general, open-ended, and multi-choice questions. For instance, IBM cooperation introduces different type of questionaires to measure users satisfaction when they interact with computer application [76].

  Like traditional questionnaire, Web-based questionnaire is a number of predefined questions designed using Web form [80, 79, 1]. Web-based questionaire is considered cost-effective method and it is easy to repeat with many users. This provides a way to collect data from large number of users. However, it is not efficient for collecting and analyzing objective data such as the efficiency of performing tasks or for measuring the numbers of errors users encounters (i.e., objective performance).

The key advantage of user-testing evaluation is the involvement of users. This allows discovering unexpected usability problems and the actual reasons that cause these problems as they encountered by real users. According to Nielsen's report[26], user-testing

[26]More information available on:`http://www.useit.com/alertbox/20000319.html`.

evaluation with 4-5 representative users will discover 80% of major usability problems related to the aspect being evaluated, while 10 users will discover up to 90% of usability problems. Note that, it is important to keep in mind and to inform all representative users that their abilities to interact with a Web site are not being evaluated. Rather, it is the ability of a Web site to accommodate to their needs and preferences are under evaluation. This is a critical distinction that should lie at the heart of any user-testing evaluation [79, 85].

However, user-testing evaluation is expensive and time consuming. Indeed, a Web site usually has many tasks and usually used by a huge number of users with different characteristics. Finding and scheduling an appropriate number of representative users for each user's type is difficult. Moreover, One or more evaluation methods could be used to evaluate one aspect of users' interactions, as mentioned before. Therefore, several evaluation methods are required in order to cover all usability aspects.

2.4.2 Usability Inspection Evaluation

Usability inspection refers to a set of evaluation methods that are performed by evaluators (i.e., usability specialists), mostly at usability design phase. In effect, Usability inspection evaluation have been proposed as alternative to user-testing evaluation, since the later is expensive and time consuming.

In practice, evaluators use a set of heuristic guidelines and usability principles to inspect usability problems relevant to a user interface design (i.e., Web page) and user's interactions with specified tasks. After that, they provide feedback for designers concerning possible usability improvements.

There are different methods that have been used for inspecting Web pages (i.e., Web sites' interfaces) and/or the interactions of users with Web sites. Heuristic evaluation and cognitive walk-through are the most common methods which have been used in Web domain [85]. The main ideas of these methods are summarized as follows:

- *Heuristic evaluation* is an informal method used mainly to inspect the layout of Web pages. Indeed, a small number of evaluators examine the Web pages of the Web site to be evaluated and judge their compliance with usability design principles (i.e., the heuristics). The output of this method is a list of usability problems in each Web page together with references to the violated heuristics [54].

- *Cognitive walk-through* is a scenario-based method that aims to simulate users' interactions. Indeed, a number of potential users' scenarios are identified. Also,

the actual conditions of use such as the types of users, objectives of use, and social and physical environments are identified for each scenario. Then, a number of evaluators simulate users to accomplish the objectives of each scenario, and discuss the usability problems as they arise [95].

Other usability inspection methods such as pluralistic walkthrough, feature inspection, consistency inspection, standards inspection, and formal usability inspection have been developed and used. A summary of these methods available on: `http://www.useit.com/papers/heuristic/inspection_summary.html`.

Usability inspection evaluation is less expensive and less time consuming. Moreover, inspection methods are considered easy to apply and useful to inspect many usability problems at design time. However, usability inspection can not predict all usability problems that could be encountered by real users such as subjective user preferences (i.e., users' satisfactions). In general, these method can can discover around 50% of usability problems [46, 54, 79].

2.4.3 Automatic Tools Supporting Web Usability Evaluation

In Web domain, several approaches have attempted to automate the process of usability evaluation or part of it. The reason can be related to a number of factors such as the need for cost-effective, real-time, and reusable evaluation. Indeed, most Web sites are rapidly changing over time. Hence, performing an exhaustive and expensive evaluation is incompatible with the rapid changing nature. Also, the evaluation results have to be accurate and up to date with respect to the current state of the evaluated Web site. Additionally, the evaluation methods have to be reused in straightforward manner when the evaluated Web site is updated or improved. Therefore, several approaches have proposed tools that support the evaluation of Web usability. These tools can be categorized into two categories, which mainly cover the aforementioned usability evaluation approaches (i.e., usability inspection and user-testing evaluations).

Automatic tools supporting usability inspection (Also called *Analytical Evaluation Tools*) are usually used to automate a combination of usability criteria and guidelines in order to analyze the design of Web pages (the user interface of a Web site). For example, several tools such as A-Prompt[27], Bobby [35], and Kwaresmi [19] have automated the W3C guidelines that are related to Web usability and accessibility (i.e., the W3C Web content accessibility guideline (WCAG) 2.0[28]). Based on these guidelines, these tools

[27] More details available on: `http://www.aprompt.ca/`.
[28] Available on: `http://www.w3.org/TR/WCAG20/`.

analyze the code of a Web page under evaluation, identify usability problems (if any), and in some cases they make suggestion for fixing the identified problems.

Automatic tools supporting user-testing evaluation (also called *Empirical Evaluation Tools*) are usually used to collect users' data relevant to evaluation when they perform real tasks. Afterwards, the collected data are analyzed by evaluators or automatically in order to discover unexpected usability problems corresponding to the tasks under evaluation. For example, Piero et al. [45] proposes a usability evaluation tool in order to analyze users' data that are stored on server log file and related to browsed Web contents, visited Web pages, and users' navigation paths. Web contents and Web pages analysis help evaluator to discover what are the browsed contents and the visited pages and how they satisfy users' needs. Navigation paths help evaluator to discover some interaction problems due to usability lacks. Additionally, WAUTER [15] is also Web usability evaluation suite that supports user-testing evaluation. This tool captures actions performed by real users and then compare them with the intended usability level based on a set of pre-defined heuristics.

In addition to the automation aspect, Web usability evaluation tools are mostly designed to conduct usability evaluation remotely. Remote usability evaluation is originally distinguished from traditional lab usability evaluation by the fact that users and evaluators do not need to be in the same location for conducting the usability evaluation. However, evaluators still have to observe users while they perform real tasks. This can be done using several tools such as video conference softwares and sensors [40].

In the Web domain, remote usability evaluation has emerged and it is mainly used to conduct user-testing evaluation online via using a Web-based evaluation tool. In this sense, users' data are automatically collected and stored during they browsing Web pages. Afterwards, the analysis process is performed automatically or by evaluators for detecting potential usability problems encountered by users. Hence, this type of evaluation is characterized by the notion that users and evaluators can be separated in time and location [110].

The adoption of automatic tools for supporting Web usability evaluation is useful for reducing the evaluation cost and the effort required by evaluators to analyze by hand the whole Web site with respect to all possible usability problems. Also, they can be reused to execute repetitive evaluation tasks over many times and highlight critical usability issues at each of these times. Additionally, Remote Web-based usability evaluation simplify the evaluation process as representative users and evaluators do not need to be in the same location, and also the existence of evaluators are no longer needed during the performance of the evaluation. However, they are not able to verify exhaustively

all usability issues. For instance, they cannot assess all those properties that require judgements by human specialists (e.g. usage of natural and concise language). Also, automatic tools cannot provide answers about the nature of discovered problems and the design revision that can solve it. Automatic tools are therefore very useful when they use as complements to the evaluation activities that are made by evaluators. since they can execute repetitive evaluation tasks for and highlighting critical features that are worth to be later inspected by evaluators.

2.5 Discussion

The primary goal of this thesis is to improve the usability of the interaction between users and Web 2.0 sites by handling the local contexts of their users, as already mentioned. Web adaptation and Web annotation are utilized as core techniques to achieve this goal. Also, we introduce an evaluation methodology which details how to evaluate the actual users' interactions after applying the Web annotation and adaptation techniques.

The works discussed above help us to build up the ideas of our thesis. However, these works have utilized Web annotation and/or Web adaptation for different purposes and in different ways. Similarly, several works have conducted Web usability evaluation to assess different usability aspects and using different evaluation approaches and methods.

To point out the differences of our work compared to these related works, the following list summarizes how our work utilizes Web annotation, Web adaptation, and Web usability evaluation for achieving the goal of this thesis.

- *Web annotation* is intended to enrich context-sensitive-contents with context information related to their authors' local contexts. Therefore, our approach does not aim to annotate document structure (i.e., document annotation) or identify the relations between contents (i.e., concepts relations annotation). It can be classified as contents annotation approach.

  To achieve this, a lightweight ontology is designed and used (i.e., local context ontology, see Section 4.5). In addition, Web annotation is designed to be interactively performed (i.e., semi-automated). An author has to specify his local context and to select Web contents to be annotated (i.e., annotation by selection user interface). Afterwards, the selected contents are annotated using RDFa-based internal annotation (see Chapter 5 for more details).

- *Web adaptation* is intended to adapt the *presentation* of annotated context-sensitive contents from their authors' local contexts to their readers' local contexts. There-

fore, our approach does not aim to provide different Web contents or different navigation links to different users, according to their preferences or skills (i.e,.personalization scenario). Also, it does not aim to adapt Web contents according to the capabilities of users' devices (i.e., mobile and ubiquitous applications Scenario). Our approach can be classified as localization approach which provides same Web contents to different users, but in different presentation (i.e., adaptation of presentation). In addition, Web adaptation is deployed at readers' applications (i.e., client-side deployment). Finally, local context ontology is used by readers in order to specify their local context information.

- *Web usability evaluation.* We introduce a Web usability evaluation methodology which explains our recommendation on how to evaluate the actual users' interaction with Web 2.0 sites after applying our usability enhancement means (i.e., Web annotation and Web adaptation). This methodology uses two user-testing evaluation methods: controlled experiment and scenario-based testing. Both of them are used to evaluate the actual result of Web adaptation and the latter is used to evaluate the actual result of Web annotation. Also, we recommend to apply these methods remotely via Web-based evaluation tools. Additionally, several usability aspects are inspected during the design of Web annotation and Web adaptation (see Chapter 7 for more details).

Part I

Web 2.0 Usability Analysis

Chapter 3

Web 2.0 Use Cases and Users Local Contexts

3.1 Introduction

As already mentioned in Chapter 1, the Web has been augmented with new features since the emergence of Web 2.0. These features allow users to contribute in creating and updating Web contents, in addition to browsing them. Also, it becomes feasible to remix and aggregate Web contents from different Web sites and presents them in a single Web page. In the meanwhile, the numbers of Web users continue to increase. These users originate from different local communities and they (need to) interact with Web 2.0 sites according to their local contexts. This leads to several semantic discrepancies, which raise new usability issues during users/Web 2.0 sites interaction. Section 1.2.1 introduces a simple example to illustrate these issues.

This chapter aims at studying and analyzing the interactions between users and Web 2.0 sites in more details, and then identifying the requirements to handle the usability problems related to users' local contexts.

The rest of this chapter is structured as follows. Section 3.2 discusses a number of Web 2.0 use cases and provides some examples for each use case. Next, Section 3.3 analyzes the types of Web contents that depends on users' local contexts with more focus on the context information used to represent and interpret each of them. Finally, Section 3.4 specifies a number of requirements to be addressed in the next chapters.

3.2 Web 2.0 Use Cases

The term *Web 2.0* has been initially introduced during a conference brainstorming session between a team from O'Reilly[1] and MediaLive[2] companies. During this session, the team members argued that the Web became more important than before, whereas new

[1] `http://oreilly.com/`
[2] `http://www.medialive.ie/`

applications and sites popping up with surprising regularity. Therefore, they realized that the Web was at the beginning of a new era.

Accordingly, the O'Reilly company has defined the Web 2.0 as *"a set of economic, social, and technology trends that collectively form the basis for the next generation of the internet which is a more mature, a distinctive medium characterized by user participation, openness, and network effects*[3]*"*.

After that, the Tim O'Reilly[4] has followed up this discussion by describing in details what is Web 2.0. Indeed, he has introduced seven principles in order to express some of the notions behind the Web '2.0' and used them as a benchmark to distinguish whether an application or approach is Web '1.0' or Web '2.0'. These principles are (1) the Web as platform; (2) harnessing collective intelligence; (3) data is the next 'Intel Inside'; (4) end of the software release cycle; (5) lightweight programming models; (6) software above the level of single device, (7) and rich user experiences. More information about these principles and the distinction are given in [89].

In Section 1.2.1, we advocate that the main idea of the Web 2.0 lies into sharing of Web contents from different users and sites. Accordingly, we explain community collaborations and contents mash-up as the main two features that characterize Web 2.0 sites.

This Section aims at discussing several use cases related to the above features which can be performed during users/Web 2.0 sites interactions. Table 3.1 summarizes Web 2.0 features and use cases, and the following subsections discuss these use cases in more details. It is worth noting that we do not aim at covering all Web 2.0 use cases, but we attempt to illustrate the above Web 2.0 features and provide some real examples related to our work.

3.2.1 Web 2.0 Contents Creation/Insertion

Most Web 2.0 sites provide one or more means that allow users to collaborate for publishing their own contents online (i.e., making them, to some extent, publicly available). Web authors now can create various types of Web contents called user-generated contents. Also, they can insert new Web contents (i.e, new html nodes) to contents which were created before. To better understand this, the following examples demonstrate the most common Web 2.0 means which support this use case.

[3]Web 2.0 Principles and Best Practices: `http://radar.oreilly.com/research/web2-report.html`.
[4]The founder of O'Reilly company.

Web 2.0 Features	**Web 2.0 Use Cases**	**Use Cases Description**
Community Collaborations	Contents creation/insertion	Creation of various types of Web contents and/or insert of new contents to contents that were created before.
	Contents update	Update of existing Web contents. Here, the update task might be performed by the original author who created this content before or by other author(s).
Contents Mash-up	Client-based aggregation	Aggregation of Web contents from several sites by user's application.
	Server-based aggregation	A Web site, via an aggregator application, aggregates Web contents from other Web sites
Collaborations & Mash-up	Contents browsing	browse of Web page's contents which might be created and updated by several authors and might be belong to several Web sites

Table 3.1: Web 2.0 features and use cases.

Weblogs

Weblogs (Also called blogs) are basically Web pages which consist of a list of Web contents that are usually displayed in chronological order with *most recent* first called *blog entries*. Weblogs are characterized by two main features. First, they usually provide a way for readers to insert *comments* about blog entries. Second, Each blog entry has a permanent URI (called permalink) which is usually described by one or two keywords. Based on the latter, blog enties are categorized and archived in a standard theme-based menu [8].

Weblogs become common after the advent of easy to use Weblogs software such as *Blogger.com and WordPress*. Indeed, Weblogs have been commonly used by individuals, but also several organizations have used them for different purposes as follows.

- *Personal Weblog.* As its name indicates, an individual author creates his own Weblog and publishes blog entries about his own personal diary, daily events, hobbies, photos, etc. Readers (or secondary authors) can only insert comments about the entries created by the original author without giving the right to create new blog entries.

- *Community Weblog.* Two or more authors create and share a Weblog page for publishing their blog entries about a specific interest or subject (e.g., football, travelers' experiences, etc.). Basically, one author creates a Weblog page and invites others to share as members [5].

- *Organization Weblog.* Several organizations and companies utilize Weblogs to achieve some specific goals. For example, Amazon.com provides a Weblog called *Amazon Dailly*[6] to facilitate the communication between books' editors/authors and books' customers. Books editors/authors can create blog entries about their books. Also, books' customers can read blog entries about books who are interested in, leave comments about a specific blog entry, or create new blog entry as a public comment to be read by other customers.

Web Forums

Web forums (also called message boards) are basically Web pages that allow a group of people to participate with ongoing and in-depth discussions about particular topics such as Web 2.0 features and technologies, new features or problems of iPhone device, and daily news. A Web forum is usually used to discuss several topics called *threads*. Each thread consists of a title, a main section called *main post*, and several discussion sections called *posts*. Threads are also categorized into a finite set of generic topics and sub-topics in a tree-like view.

Web forums are usually governed by several persons called *forum staff*. Also, most forums require users registration to participate in a discussion. In practice, forums can be categorized into two types according to permissions given to registered users (i.e., members) by forum staff as follows.

- *Discussions and replies forum.* Members can create new threads (i.e., new titles and main posts) for discussions, and also can reply on existing threads by inserting posts. This type is commonly used in most forums. for example, BBC news site provides a discussion space for their members called *My Discussions*[7]. Members can create new thread for discussions or post messages about existing topics such as daily news. O'Reilly company also provides a forum that allows members to create new threads for reviewing books, discussing a set of already mentioned topics, propose new topics for discussions. Also, they can reply to existing discussions.

[5] See group blogging feature on `http://www.blogger.com/features`.
[6] `http://www.amazon.com/gp/daily`
[7] http://www.bbc.co.uk/messageboards/newguide/

- *No discussions but replies forum.* Members can only insert new posts for discussing existing threads. This type is commonly used for supporting discussions between teachers and students in E-learning applications. Students are usually given permissions to reply on discussions which are created by teachers[8].

Web 2.0 Advertising

Several Web 2.0 sites provide a way for individuals and companies to advertise their products for selling. For example Google provide an advertizement service called AdWords (or ads for short). This service allows advertisers to create their advertisements contents (e.g., New and second hand cars for sale, starting from € 500) and choose keywords or phrases related to their products or services (e.g., used cars). When people search on Google using one of the chosen keywords, ads may appear in the search results. Recently, social network sites such as FaceBook provide an advertising service for its members like the Google AdWords service.

In addition, eBay is one of the most famous site that provides different advertizement services, in addition to products selling and buying. For instance, eBay provide a service called *adCommerce*[9], whereas advertisers (i.e., individuals or companies) can create advertizement contents about their products (e.g., MP3 Player) and display them for international customers, a specific target customers (e.g., young people), or for a specific demographic area (e.g., Belgium).

Web 2.0 Social Networking

Web 2.0 social networking focuses on building and reflecting social network or social relationship among people. Basically, social networking sites (SNSs) such as FaceBook, MySpace, and LinkedIn allow users to create public or semi-public profiles (e.g., name, photo, birthdate, email, phone number, etc.) for representing themselves as members, and communicate with other members who have common (social) relations such as friendship, shared interests, professions, etc.

In addition to members' profiles, most SNSs allow members to create various types of contents such as describing upcoming events, asking and answering questions, sending messages or comments to other members. Furthermore, one member can create a common page usually called *group*. The creator of a group can invites any one to join,

[8] More information available on: `http://docs.moodle.org/en/Forum_module`.

[9] See eBay advertisement services on `http://www.ebayadvertising.com/en/`, last visit on 20/12/2009.

deny join request, and delete or update the contents of joined members. Joined members usually can browse the contents exist on the group page and create new contents. Finally, other additional Web 2.0 features such as Blogs sharing, Web 2.0 advertising, and forums discussions are provided by some SNSs [25].

3.2.2 Web 2.0 Contents Update

In addition to the above creation/insertion use case, Web 2.0 sites usually enable Web authors to *update* Web contents after they were published online. Indeed, a Web author could update Web contents that s/he created before (referred to as personal contents update). Also, s/he could update Web contents that were created by other authors (referred to as community contents update).

Personal Contents Update

To better understand personal contents update, let us consider the Web 2.0 means described above. Indeed, most of them allow an author to update Web contents s/he created before. For instance, Weblogs usually allow authors and commenters to update their own blog entries and comments, respectively. Also, Web 2.0 advertisement services and sites such as Google AdWords and eBay allow an advertiser to update Web contents related to item(s) s/he wants to advertise such as its price, photo, and location. In contrast, other users can not update these contents. The same case for Web 2.0 social network and Web forums.

Community Contents Update

In general, the Web 2.0 means described above do not allow an author to update Web contents created by other authors. However, a few persons can update other authors' contents. For instance, Weblog owners (i.e., Weblogs creators or organizations) can update blog entries and comments created by other members. Also, the forum staff can update posts created by the members of the forum. Finally, the creator of a group in SNSs can update contents created by joined members. These provide examples about community contents update. In addition, Wiki and collaborative Web editing tools are the most common Web 2.0 means which provide straightforward examples about community contents update.

Wiki

A wiki is a set of interlinked Web pages whose contents are collaboratively produced by its (authorized) users[10]. *Wikipedia* is the most famous wiki site. Wikipedia is a public wiki that allows any user to create, insert, and update Web contents about any things. For example, a Web author can define and publish the term *local context* on wikipedia. Also, other author can read and insert new definition of this term.

The main distinctive feature of wikis among other Web 2.0 means described above lies into *open editing* notion. By open editing, we mean that community contents update are not restricted to a few users such as Weblog's owner and group's creator in SNSs. However, any user can update contents created by other users. For example, any user can update the definition of the term *local context* which is already created by other author(s) on wikipedia.

Collaborative Web Editing Tools

Collaborative Web editing tools are Web applications which provide an online space and instruments for (authorized) users to create, share, and update various types of documents such as XHTML, text, speed sheet, and presentation documents. One user can create and upload shared document. Then, co-authorized users can insert new contents to this document and/or update the existing ones. In other words, collaborative Web editing tools is considered like traditional disk-top office applications, but with additional open editing, collaborative, and online features.

Several Several collaborative Web editing tools has been proposed and used[11]. For example, Google provides a number of collaborative edition tools called *Google Docs and Google Wave*[12]. These tools allow users to collaborate to create and update almost any type of documents such as text document, Web form, presentation, speed sheet, photo, video, map, and more. Acrobat.com also provides a collaborative Web editing tools called *Online Office Applications*[13]. Finally, wiki sites can be considered as a special kind of collaborative Web editing tool.

[10] Some wikis require user's authorization to read, create, or update wiki contents. These are called private wikis

[11] See some list on `http://en.wikipedia.org/wiki/Collaborative_real-time_editor`

[12] See `http://docs.google.com/` and `http://wave.google.com/` for more information.

[13] `https://acrobat.com/features_online_office_applications.html`

3.2.3 Web 2.0 Contents Aggregation

Contents aggregation refers to a process of finding out and aggregating Web contents from several Web sites and display them together. This process is collaboratively performed as follows. First, users (i.e., contents' authors or providers) need to publish Web contents to be aggregated using machine readable data format. Also, they have to agree and announce that these contents can be aggregated. Second, a specific type of applications called *aggregators* find out and aggregate these contents. Finally, Web readers browse the aggregated contents. In effect, contents aggregation embodies the Web 2.0 contents mash-up feature.

There are two types of contents aggregation. First, aggregation of contents that share a similar data format such as blog entries, news entries, or events (referred to as data aggregation). Second, aggregation of contents that have different data formats, but they have common relations or they complement each others. This type of aggradation is commonly provided as Web services (referred to as service aggregation). From architecture perspective, an aggregator could be deployed on a user's device. This kind of aggregation is referred to as client-based aggregation. Also, it could be deployed on an intermediary Web server or on a server which host a Web 2.0 site. This kind of aggregation is referred to as server-based aggregation.

Client-Based Aggregation

Client-based aggregation has been commonly used for aggregating Web contents that share a similar data format (i.e., data aggregation). Then, the aggregated contents could be stored on user's device, exported to another user's application, or even exported to another Web site. The following provides some examples about common aggregators deployed on users' devices.

Web Feeds Aggregator

A Web Feeds aggregator (also known as feeds reader or news reader) is a well-known aggregator that extends most Web browsers[14]. A feed aggregator is used to aggregate *Web feeds* (also called syndicated Web contents) from several Web sites periodically.

Web feeds are basically machine readable data formats such as *Really Simple Syndication*[15] *and Atom Syndication Format*[16] *(RSS and Atom for short)*. These formats are

[14]Web Feeds Aggregator are also used as a server-based aggregator.
[15]`http://www.rssboard.org/rss-specification`
[16]`http://tools.ietf.org/html/rfc4287`

used for *syndicating frequently updated* Web contents or summary about these contents. In practice, Web feeds are almost utilized to syndicate the contents of News websites, Weblogs, and forums. Also, they are utilized to syndicate other types of Web contents such as weather data, currency exchange rates, and top-ten lists of hit tunes to search results.

In a typical scenario, a Web feed aggregator works as follows. a Web reader subscribes to one or more feed-enabled Web sites[17]. Accordingly, the aggregator periodically monitors these sites in order to find out and aggregate Web feeds that are recently updated and inform the reader thereafter.

FireFox Operator: Microformats and RDFa Aggregator

As its name indicates, *FireFox operator*[18] is an extension for firefox Web browser. It is used for aggregating Microformats specifications and/or RDF statements which are embedded into Web pages using Microformats and RDFa languages, respectively. In addition, it allows users to export the aggregated information into other disk top applications or other Web sites. For instance, FireFox Operator allows user to export Microformats-based or RDRa-based *events* from a Web page to Microsoft outlook, Google calendar, or Upcoming Web site.

Piggy Bank

Piggy Bank is another extension to FireFox Web browser that allows a Web user to aggregate Web contents from different Web sites and save them on his computer using RDF data model. After that, the user can browse the aggregated contents and also share them with other users [60].

Server-Based Aggregation

Server-based aggregation has been used for aggregating Web contents that both share similar data formats and related contents that have different data formats (i.e., data aggregation and service aggregation). In practice, server-based aggregation is performed by server aggregators that work as proxies between published Web contents to be aggregated and Web users. The following list provides some example about common server-based aggregators.

[17] Web sites that syndicate Web contents as Web feeds

[18] `https://addons.mozilla.org/en-US/firefox/addon/4106`

Data Aggregators

Tecnorati[19] *and Upcoming*[20] are two common Web 2.0 sites which provide data aggregation means. Tecnorati has lunched as Weblogs aggregates, whereas blog entries are aggregated and indexed from many Weblogs pages. Subsequently, other types of Web contents are also aggregated and indexed such as photos and news. Users can search and browse the aggregated contents according to their types (e.g., blog entries), language, published/updated time, etc. Upcoming aggregates events contents from different communities of users and also from commercial sites. The purpose is to let users to discover events who are interested in and/or to share these events with their friends.

Service aggregators

Several service provided by Google such as *AdWords, AdSense, and Google maps* provide good examples about service aggregator. Google AdWords is an advertisement service which allow Web providers (e.g., Web site owner) to add text, image, or video advertisements to other Web sites that mostly provide related Web contents/services. Google AdSense is complement service to the AdWords. The sites owner give permissions to Google to add advertisement contents to their Web sites when they appear in the google search result. Google also provides a Web mapping service known as Google maps. This service allows Users/Web site owners to embed maps related to specific locations inside their Web pages' contents using Google maps API.

Booking travel packages provided by travel agencies is another common service aggregation example. Many travel agencies such as *Expedia*[21] *and eDreams*[22] aggregate (compose) travel reservation services (i.e., flight booking, hotel reservation, car rent) from different Web sits or service providers to satisfy users' requests

3.2.4 Web 2.0 Contents Browsing

The traditional scenario of browsing a Web page's contents can be summarized as follows. A Web user requests the intended Web page, whereas the latter belongs to a Web site and could be stored as a static HTML document or might be generated dynamically from the data sources of this site (e.g., site's database or XML documents). Then, the requested page is rendered via HTTP protocol and presented on this user's application

[19] `http://technorati.com/`
[20] `http://upcoming.yahoo.com`
[21] `http://www.expedia.com/`
[22] `http://www.edreams.com/`

(e.g., Web browser) for browsing.

Although the basic idea and the utilized technologies are still the same, the Web 2.0 use cases described above lead to new browsing scenarios. For instance, the aggregator engines that extend users' applications introduces new browsing scenario as follows (e.g., client-based feeds aggregators). Instead of requesting an entire Web page by a user, the aggregator engine acts on behalf of this user to request and aggregate specific types of Web contents from many Web sites (i.e., RSS feeds). Then, the aggregated Web contents are generated and presented as a Web page on user's application. Finally, the aggregator usually notifies this user to browse the aggregated contents. Additionally, server-based aggregation use case leads to similar scenario.

In addition, the participation of Web users in Web contents creation/insertion and update leads to another browsing scenario. A Web user can browse Web page's contents, and then insert new contents such as personal comment. Also, he can update or delete what he browsed, or recommend them to other users (see contents creation and update use cases for more information).

Finally, many other browsing scenario are also possible, including a mix of the above browsing scenarios. For instance, a Web user can request a Web page using the traditional browsing scenario. Afterwards, he can utilizes an aggregator engine to aggregate and export part of the browsed contents such as event contents on the Web. Finally, another user can insert or update the exported contents.

3.2.5 Web 2.0 Use Cases: Summary

The aforementioned use cases show that users can do more than just browse Web contents. In addition, the roles of Web sites and users' applications become more than just hosting and presenting these contents, respectively. Indeed, Web users participate in the production of Web contents, in addition to their traditional browsing roles. Consequently, users' applications are extended to allow users to create and update Web contents and also to aggregate contents from different sources. In addition, several Web 2.0 sites act as Web contents aggregators, in addition to their traditional roles (i.e., hosting Web pages and rendering them to users). Finally, the ways users browse Web contents are also changed accordingly, as already mentioned. Figure 3.1 presents a simplified model that summarizes the aforementioned use cases and the new roles of the Web entities.

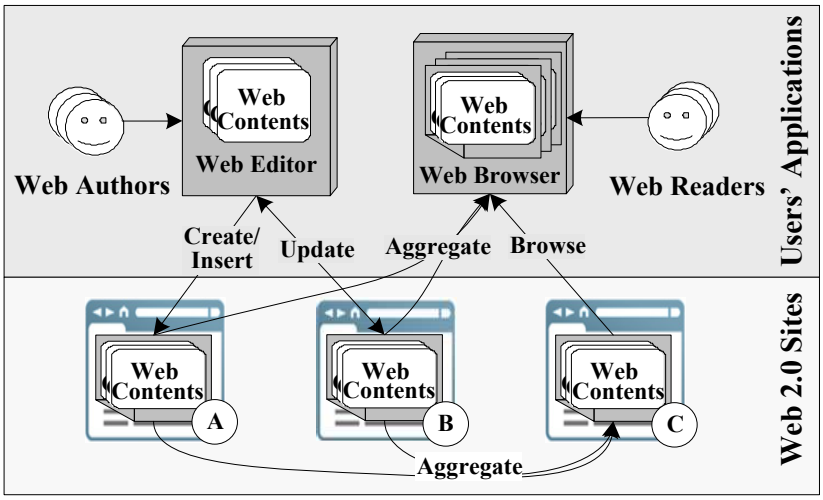

Figure 3.1: Web 2.0 use cases

3.3 Local Context and Context-Sensitive contents ($CSCs$)

Context-sensitive contents ($CSCs$) are defined in Chapter 1 as particular types of real-world concepts which are represented and interpreted in different ways by different Web authors and readers, respectively. Also, Section 1.2.1 defines what is meant by local context, and provides an example which illustrates how several date and length contents are represented and interpreted in different ways. This section describes several types of $CSCs$ and the types of context information that are implicitly used to represent and interpret each of them.

3.3.1 Date and Time

In *Longman* dictionary[23], the *date* as a noun refers to a particular day of a month or a year within a calendar system. In this sense, the definition of the Date itself depends on the local contexts of users. Indeed, users may use different calendar systems such as Gregorian, Islamic, Japanese, and Chinese calendars. These calendars count the number of a year days in different way. For instance, the year in Gregorian calendar includes 365/366 days, whereas the year in islamic calendar includes 354 days. Hence, a particular day in one calendar may be represented by a different date in another calendar[24].

Even though the same calendar system is used, different styles and different conventions are used to represent dates. For example, The representation of a numeric date in

[23] `http://www.ldoceonline.com/`

[24] See more information about calendar systems on `http://en.wikipedia.org/wiki/Calendar`.

Gregorian calendar could start with date, year, or month and with different separators between them, as already illustrated in Chapter 1. Moreover, the textual representation a Date in the same country might be represented in different ways, whereas the name of the dates, months, and years are represented in a local langauge.

In addition, the *date* term are also considered as a time duration of 24 hours between two successive midnights. This also depends on the local contexts of users. Indeed, the time is usually interpreted according to the local time of the region (or time zone) that is used it. Also, it might be represented as 12-hour clock or 24-hour clock and with different separators.

In order to unify the date/time representation, The ISO proposes an international and unambiguous representation of date/time in a standard called ISO 8601[25]. This standard is based on Gregorian calendar system and organizes the date/time from the largest to the smallest temporal terms (i.e., from year to second). Also, it optionally attaches the time zone offset after the date/time representation.

Gregorian calendar is used by most countries wide world as it is considered the de facto international calendar. However, most countries still represent date/time contents using local date/time conventions. This work considers date/time $CSCs$ as a Gregorian calendar date/Time, with different local writing formats and different time zones conventions [64]. Therefore, the following context information is related to date/time $CSCs$:

1. The writing format in which a date or time is represented. This includes the style used to represent them (i.e., text or numerical). Also, the order that a year, month, day are organized, and also wheatear 12-hour clock or 24-hour clock time format is used. Finally, the punctuation used to separate dates and times elements

2. The time zone convention used to represent the value of a date or time

3.3.2 Numbers

In mathematics, *numbers* are mainly used to count objects or measure their quantities based on a number system. Base-ten system is the most common number system, whereas numbers are represented as combinations of ten digit number symbols called numerals. The most common used numerals is called Hindu-Arabic numerals (i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Most countries such as most Europe countries and USA[26] use

[25] `http://en.wikipedia.org/wiki/ISO8601`
[26] United States of America

these numerals to represent numbers. However, many countries still use their own local numerals. For example, most arabic countries use eastern arabic numerals[27].

In addition, countries or communities that use the same numeral system could use different formats to represent numbers. To illustrate this, let us discuss what does the number $1,234$ represent the USA and France? In USA, this number represents one thousand two hundred and thirty four. However, it represents one and two hundred thirty four one-thousandths in France. This confusion occurs since numbers are by represented using different decimal and thousand separators in the aforementioned two countries. Indeed, USA use *dot* as a decimal separator and *comma* as thousand separators, whereas the *comma* is used as a decimal separator in France [64]. To conclude, *numbers CSCs* are represented and interpreted based on the following context information:

1. The notational symbols used to construct numbers (i.e., numerals)
2. The types of separators used to represent decimal and thousands separators

3.3.3 Telephone Number

Numbers are also used to represent telephone numbers, in addition to counting and measuring roles. A *telephone number* refers to a unique sequence of decimal digits that identify a telephone end-point (also called a telephone line) in a telephone network. Using this number, a user can make a telephone call (i.e., to dial) to this telephone line from other lines.

More technically, a telephone number refers to the definition of the term national (significant) number (or N(S)N) provided by the International Telecommunication Union (or ITU for short[28]) in the international numbering plan E.164. N(S)N involves a subscriber number and a national distinction code as two basic entities. A *subscriber number* identifies a telephone end-point on a telephone network, and a *national distinction code* identifies a telephone network among other telephone networks used in a country or groups of countries[29].

To make a telephone call, a telephone number is usually complemented with one or more prefixes. These prefixes depend on several conventions related to the local context of a user who want to make this call. In addition, they are related to the telephone number that this user wants to dial. Indeed, one prefix is needed to dial a local telephone number (i.e., from inside a country). This is called *national prefix* and

[27] More information available on `http://en.wikipedia.org/wiki/Numeral_system`.
[28] http://www.itu.int/
[29] More details available on: `http://www.itu.int/rec/T-REC-E.164-200502-I/en/`

precedes the dialed telephone number. Most countries use 0 digit as national prefix. However, some countries use other code. For instance, USA use 1 as national prefix.

This national prefix is not needed to dial an international telephone number (i.e., from outside a country). Instead of this, two other prefixes are needed: international prefix and country calling code. *International prefix* (Also called international access code) involves two or three decimal digits and identifies the source country which the telephone number is called from. Here, different countries also use different international prefixes. For instance, USA and Canada use 011, while most countries in Europe use 00 as international call prefixes. *country calling code* is another two or three decimal digits that identifies the destination country corresponding to the dialed telephone number. In practice, each country has a unique country calling code. For example, the country calling code for Belgium is '32' and for France is 33[30].

To better understand the local conventions related to telephone numbers, let us consider a Belgian telephone whose number is 81123456. The following scenarios have to be followed to dial this telephone. First, users from inside Belgium should dial this telephone by adding the national prefix used in Belgium (i.e., 0). Hence, the dialed number should be 081123456. Second, users from outside Belgium should omit the national prefix and add the international prefix and then the country calling code instead. Here, the international prefix depends on the country that a user dial this number from. For instance, users from USA should add 011 as international prefix , and 32 as Belgium country calling code, and then the telephone number itself. Hence, the dialed number should be *011 32 81123456*. In contrast, users from France should add 00 as international prefix, and 32 as Belgium country calling code, and then the telephone number itself. Hence, the dialed number should be *00 32 81123456*.

In addition, different writing formats are used for expressing telephone numbers in different countries or communities. For example, telephone numbers in France are written as pairs of digits with spaces between pairs (i.e., dd dd dd dd), whereas the common format in USA is written as two triples and one 4-tuple of digits (i.e., ddd ddd dddd). To conclude, the following context information is related to *telephone number CSCs*:

1. The national call prefix (for telephone calls inside a country)
2. The international call prefix related to source country
3. The international calling code related to destination country

[30] See the list of country calling codes on: `http://en.wikipedia.org/wiki/List_of_country_calling_codes/`

4. The local writing format used to represent the value of a telephone number

3.3.4 Physical Quantities

Physical quantities such as weights, lengths, and temperatures are measured using measure units such as Gram, Meter, and Celsius, respectively. Measure unit is a reference value that are agreed upon and practically used by community of people to measure a specific kind of physical quantity (called a quantity dimension). Also, measure units may be combined with prefixes. These prefixes are also agreed symbols that precede measure units to represent multiple or fraction values of these units. Finally, a combination of a number value together with (prefixed) measure unit is used to represent a corresponding quantity dimension. These elements form what is known measurement system.

The most common and widely used measurement system is called international system of units (abbreviated as SI). SI is a revised version of the Metric measurement system and it is devised around seven base units. Then, other units are derived from the base units. In addition, several unit prefixes are proposed as multiples or fractions of the number 10. For example, the prefix *kilo* refers to thousand (i.e., 3 * 10)[31].

However, some countries still use other measurement systems. For instance, USA and United Kingdom use Imperial system or its derived version, and weight quantities are measured using Pound and Ounce units. Furthermore, different countries represent the value of physical quantities in different ways (see *number CSCs* above). Finally, unit symbols and prefixes could precede quantity values or they could succeed them[32]. To sum up, the following context information is related to *physical quantity CSCs*:

1. The quantity dimension which has to be measured
2. The local measure unit used to measure a quantity dimension
3. The measure prefix that might be combined with a measure unit
4. The local writing format used to represent the value of a physical quantity

3.3.5 Price

The term *Price* refers to a numerical monetary value assigned to a good, service or asset to exchange them between sellers and consumers (e.g., peoples, organizations, etc.). Each price CSC is usually represented using a number value together with a currency,

[31] More information about SI system are available on `http://www.bipm.org/en/si/`
[32] More information available on `http://en.wikipedia.org/wiki/Units_of_measure`.

whereas the letter is a unit of exchange. In this sense, price $CSCs$ can be considered as a special kind of physical quantities. Indeed, there are two aspects that distinguish price $CSCs$ from other physical quantities: the dynamic nature of currency units and the local conventions related to sales tax.

In effect, many currencies are used in different countries to exchange goods and services[33]. The value of each currency is determined based on its rate among other currencies (called currency exchange rate). This rate is frequently change based on several economic aspects related to the country or countries that issue and/or use this currency. Conventionally, The U.S. Dollar is considered as a reference currency and the rates of other currencies are calculated accordingly. However, the value of this currency is also changed frequently.

In addition to currency aspect, the price value could include or exclude the value of sales tax. Basically, a sales tax is a kind of a consumption tax imposed as a percentage of the price of goods or services. Several tax systems are used in different countries to apply it. The most common used system is called *value added tax* system (or VAT for short). Using VAT, the sales tax value is paid by an end-consumer and it is collected by a seller. Also, a user who is in the middle of selling-consuming chain has to pay sales tax, but a part of this sales tax can be deducted when (s)he sell the product or service who bought before.

VAT is used by large numbers of countries all over the world. For instance, most Europe countries use this sales tax system[34]). However, several counties still use different sales tax systems. For example, Australia use *Goods and service* sales tax system (or GST for short). Moreover, different tax rates are usually used in different countries even though they use the same tax system. Finally, one country might use different tax rate for different types of goods or services[35].

In practice, a sales tax is traditionally applied on selling/buying transactions which are occurred inside a country. Therefore, it is paid to this country and calculated according to the sales tax system of this country. However, the Web (e.g., via e-commerce, e-service, and e-marketing sites) provides a means for sellers to sell goods and services for buyers from different countries, and vice versa. This new scenario raises the following challenging questions. First, which sales tax system have to be applied: the system applied in a buyer country or the system applied in a seller country? In addition, to which of these countries this tax have to paid? Indeed, there is a strong debate regarding

[33] See ISO 4217 for currency list and corresponding currency code.

[34] `http://ec.europa.eu/taxation_customs/taxation/vat/index_en.htm`

[35] To have a look about countries tax rates, see: `http://en.wikipedia.org/wiki/Tax_rates_around_the_world`

to the right answers for these questions. On one hand, it is considered from legislation perspective that the sales tax have to be paid to a buyer country, and according to the sales tax system applied in this country, since this tax is paid by a buyer. On the other hand, applying this in practice is considered very complex. For instance, assume the sellers who are responsible to collect the sales tax, how they can calculate the sales tax according to different sales tax systems. Also, how can sellers pay the collected tax to different buyers' country.

From users' perspectives, it is important for them to know the following aspects when they sell/buy goods or services on the Web. First, whether the sales tax is included or excluded in the price values of goods or services. If so, they want to know the type of sales tax system and the tax rate that are applied on the price of goods and services. From sellers' perspectives (i.e., authors), they need this to be able to add the value of sales tax and to collect them correctly from buyers (i.e., readers). From buyers perspective, they need to know how much they pay tax when they buy a good or service and also they need this to be able to reimpose the amount of sales tax if they are mediator users of these goods or services.

Finally, different countries represent the value of price $CSCs$ in different ways. Also, currency signs could precede price values or could succeed them, like the representation of physical quantity $CSCs$ discussed above. To conclude, the following context information is related to price $CSCs$:

1. The local currency used to exchange a product or service
2. Whether the sales tax included or excluded, the sales tax system used, and the sales tax rate.
3. The local writing format used to represent the value of a price

3.3.6 Context-Sensitive Contents: Summary

Table 3.2 below summarizes the aforementioned context-sensitive contents and their corresponding context information. These cover the most common $CSCs$ that almost appears in all of the aforementioned Web 2.0 use cases.

3.4 Requirements of Improving Web 2.0 Usability

As already stated in Section 1.1, usability analysis aims at studying and analyzing the effect of several aspects on the usability of users' interaction with Web sites. According to

Context-Sensitive Contents	Local Context Information
Date/Time	Time zone, date/time format
Number	Numerals, number format
Telephone Number	Country calling code, international prefix, national prefix, phone format
Physical Quantity	Quantity dimension, measure unit, unit prefix, quantity format
Price	Currency, sales tax included/excluded, sales tax system, sales tax rate, price format

Table 3.2: Relations between context-sensitive contents and local context information

the result of analysis, it also aims at specifying the requirement to address the potential usability problems. In this sense, Web 2.0 use cases and the users' local contexts used to represent and interpret *CSCs* are the main aspects that affects on users/Web 2.0 sites interactions.

Prior specifying the requirements of improving Web 2.0 usability, we have to emphasize on the following aspects. First, Web users act as active Web authors who can create and update Web contents, according to the aforementioned Web 2.0 use cases. Also, they practice new browsing scenarios, in addition to the traditional browsing scenario.

Secondly, it is clear that the local context information is part of the *CSCs'* semantics, since Web users implicitly use their local contexts to represent and interpret these *CSCs*. This proves that the representation of *CSCs* are *incomplete*, as we described in Section 1.2.2. Moreover, these *CSCs* are very frequently appear in all the Web 2.0 use cases as parts of Web contents that have predefined schema such as start and end dates in calendar events (i.e., structured Web contents). Also, they could be also parts of semi-structured/unstructured contents such as price contents inside Weblog entries. In addition to *CSCs* described above, there are other types of Web contents that depend on users' local contexts such as personal naming conventions (e.g., person name may start with first name or family name. Also, father name could be also included) [116], notations and symbols for representing mathematical formulas and equations [83], and geographical addresses.

Thirdly, since the representation of *CSCs* are not complete, Web users could encounter several discrepancies when they interpret these *CSCs*, and consequently they require additional efforts to interpret these contents or even misinterpret them (i.e., inefficiency or inaccuracy problems).

To improve Web 2.0 usability, there is a need *to complete the representation of the $CSCs$ with their authors' local contexts and to adapt them according to their readers' contexts.* In addition, the proposed solution to address this requirement has to consider the aforementioned Web 2.0 use cases. To this end, the following interrelated issues have to be tackled.

1. **Semantic Information Identification and Representation**

   The first issue is related to the information required to identify the semantics of Web contents (i.e., the type of $CSCs$) and the local context of Web users. As it is shown above, Web contents in a single Web page might be created, updated, and aggregated from different sources (i.e., authors and sites) and might involve several types of $CSCs$. This implies that different Web contents from different sources could refer to the same type of CSC. For example, different authors could use *cost*, *price*, and *amount* contents to refer to the *price* concept. Therefore, the following question has to be answered: what is the information required to identify the semantics of a CSC, such that user's application can interpret the type of this CSC?

   In addition, each type of $CSCs$ is represented according to different types of context information, As it is shown above. Therefore, the relations between $CSCs$ and their corresponding context information have to be represented at conceptual (meta) level. This representation must be must inter-operable and flexible among users' applications. More precisely, it has to consider that values of $CSCs$ from different sources are represented according to different authors' local contexts, might be aggregated by other Web sites or users' applications, and need to interpreted from different readers' local contexts. Recall that, Section 2.2.2 discusses several critica which need to be considered to select a suitable representation technique. These are related to domain applicability, inter-operability, extensibility, context model decoupling, and reasoning capability of the selected technique.

2. **Local Context Management**

   In addition to the aforementioned issue, the local context information have to be acquired and stored. Also, a suitable context information has to be associated with each corresponding CSC at run time. Thus, the following questions have to be addressed. First, how context information be acquired? More specifically, has to be acquired directly from users or has to be acquired (predicted) by users' applications? Second, When this information has to be acquired (i.e., before or

during *CSCs* creation, update, and browsing)?

In addition, the local contexts of both authors and readers have to be accessible by applications that perform the adaptation process. Accordingly, we should consider how and where these local contexts are stored (i.e., on users' devices, on Web site data store, or both). Finally, the association of local context information with *CSCs* should be hidden from the authors. In effect, we have to consider that authors are non experts and often do not know the relations between *CSCs* and local context information.

3. *CSCs* **Adaptation**

   The ultimate requirement of our approach is to adapt *CSCs* to their multiple readers' local contexts in all Web 2.0 use cases. In addition to the above issues, we need to deal with where and when the adaptation of *CSCs* have to be applied. Indeed, this process can be performed at creation/update time, at aggregation time, or at browsing time. Also, it can be deployed on users' applications or on servers that host Web 2.0 sites. Also, it can be deployed on intermediary servers (see Section 2.2.4).

   Having considered the above issues, the *CSCs* have to be adapted according to readers' contexts. This implies that Web readers have to specify their local contexts. In addition, a Web page to be adapted have to parsed in order to locate *CSCs* from the entire Web contents involved in this page. Finally, the type of each *CSC* has to be identified and the appropriate adaptation has to be performed.

Part II

Web 2.0 Usability Design

Chapter 4

Semantic Representation Model of $CSCs$

4.1 Introduction

To enhance the usability of users/Web 2.0 sites interactions, Section 3.4 identifies two requirements: First, completing the representation of $CSCs$ with their authors' local contexts; second, adapting these $CSCs$ according to their readers' contexts. Also, it raises three issues to be tackled in order to address these requirements: semantic information identification and representation, local context management, and $CSCs$ adaptation.

This chapter mainly aims at tackling the first issue. To this purpose, it initially discusses several design alternatives and evaluate them with respect to Web 2.0 use cases. As we will see, our evaluation is based on several design principles/criteria such as do not repeat yourself principle (known as DRY).

Next, it introduces a semantic representation model. This model is basically based on the notions of *semantic object and local context ontology*. Semantic object allows authors to enrich (i.e., complete) the representation of $CSCs$ with semantic metadata to facilitate their automatic adaptations. This semantic metadata describes the underling semantics of $CSCs$ and provides an explicit description about their authors' local contexts. Local context ontology provides a mean for representing local contexts information, so that users (authors and readers) can specify their local contexts. As will see in Chapter 6, it becomes feasible for readers' applications to adapt $CSCs$ from their multiple authors' contexts to their multiple readers' contexts by utilizing this model.

Finally, an architecture is introduced at the end of this chapter. This architecture assembles the components of our approach and illustrates how it works seamlessly with the existing Web technology stack.

The rest of this chapter is structured as follows. Section 4.2 evaluates three design alternatives. Section 4.3 discusses the semantic object notion and its related aspects. Section 4.4 presents some common ontologies to be reused. Section 4.5 describes the

design of the local context ontology. Section 4.6 presents typical representation of *CSCs* as semantic objects. Finally, Section 4.7 introduces our proposed architecture.

4.2 Design Alternatives

There are several design alternatives to address the Web 2.0 usability requirements identified in Section 3.4. Each of these alternatives relies on different ways to complete the representations of *CSCs* with their authors' local contexts and to adapt them according to their readers' contexts. This section identifies three design alternatives and evaluate them based on the following design principles/criteria:

- *Context representation flexibility.* The extent that context representation can be extended to satisfy all Web sites.
- *Do not repeat yourself (DRY).* The extents that the design alternative to be evaluated comply with DRY principle [3]. This principle is discussed in more details in Section 2.3.3.
- *Number of CSCs adaptation.* How many times *CSCs* have to be adapted during their life cycle (i.e., creation, update, and browse).
- *Inter-operability.* The extent that the representations of *CSCs* together with their authors' local contexts are inter-operable among users' applications and Web sites, so that the former can be adapted according to their readers' contexts.

4.2.1 Adaptation to a Standard Local Context

The first alternative imposes a standard, unified local context for all Web sites. Then, each *CSC* needs to be annotated with a standardized machine interpretable version (referred to as *MV*). The latter is generated by adapting the value of *CSC* from its author's local context to a standard context at creation and update time. Additionally, there is a need to adapt the *MV* into different human-readable versions according to the different readers' contexts.

In practice, we can rely on Microformats[1] for embodying this alternative. As already mentioned in Section 2.3.3, Microformats propose a set of standards, called *specifications*, and reuse XHTML attributes such as *id* and *class* to embed these specifications into XHTML documents. Indeed, Microformats specifications standardize the representation of Web contents at different three levels as follows:

[1]See `http://microformats.org/wiki/`

- *Schema level.* Each proposed specification has a specific schema identified by a main concept and a set of sub-concepts (called class and subclasses). In addition, the cardinalities (e.g., required, optional, many) and the ordering of the subclasses for each schema are also identified. For example, $hCard$ specification is identified by $vcard$ as main class, fn and n subclasses at minimum (i.e., required classes), and a set of optional subclasses such as $email$, $photo$, and adr. In addition, a subclasses could have other subclasses. For instance, adr subclass has a set of subclasses such as *postal-code and country-name.*

- *Concept level.* A specific vocabulary (i.e., Semantic label) is dedicated for each class and subclass involved in Microformats specifications.

- *Representation level.* A specific representation is identified for the value of each class and subclass. Authors should follow these representations as much as possible, so that Microformats parsers can interpret these values. However, if some Web contents are not interpretable from either human or machine (e.g., $CSCs$), then authors need to provide two versions of these contents: one for human and one for machine (i.e., MV). In this context, the authors have to follow Microformats *design patterns*, to ensure correct interpretation of these MVs.

Accordingly, the local context information used to represent $CSCs$ is standardized at one or more of these levels. To better understand this, the following gives an insight on the way Microformats standardize the local context information of the *date/time and length contents.*

With respect to date/time contents, Microformats community proposes a design pattern called *date/time design pattern.* This pattern recommends to annotate a date/time content with an MV based on the numeric ISO 8601 date/time standard. Hence, the local context information related to date/time contents is standardized at representation level as follows. The date style is always short (i.e., numeric), and the date/time format is from most to least significants (i.e., from year to second). Also, the time zone is included in the MV value.

With respect to date/time contents, Microformats community proposes an $hMeasure$ specification with num and $unit$ as required subclasses, and $type$ and $tolerance$ as optional subclasses. In addition, it recommends to use an *abbr design pattern* to annotate the length contents (physical quantities in general) with MV numerical amount and MV unit code. The MV numerical amount is based on a specific free-context grammar pattern (called Extended Backus-Naur Form, or EBNF pattern) and with scale factor

equal 1. Also, the MV unit code is based on the official unit codes of the international system of unit (SI) or other unified units codes for other non-SI units[2]. Therefore, the local context information used to represent length contents is standardized at schema and representation levels.

Example

To better understand this alternative and its consequences on the Web 2.0 use cases, Figure 4.1 illustrates the adaptation of the date $CSCs$ presented in Section 1.2.1 using this alternative. Here, The date contents in pages A and B are annotated with suitable MVs. Then, they are aggregated to page C.

In page A, an MV version is firstly generated from the date CSC updated[3] by the American author at task T3. Secondly, the generated MV and the date subclass from the *vCalender* specification are embedded inside the XHTML tag used to represent this $CSCs$ (i.e., abbr tag). Note that, this is performed using the *abbr design pattern*.

Similar scenario is performed for generating and embedding an MV for the date CSC created by the Canadian author in page B. Next, these dates and their corresponding MVs are aggregated like *"copy and past"* from pages A and B to page C (i.e., Task T5). Finally, The MVs are adapted according to the French reader's context at Task 7. Note that, the length $CSCs$ are adapted in similar way, as it is partially shown in this example.

Discussion

This alternative allows $CSCs$ from several sites to be aggregated seamlessly as they are annotated with unified MVs. However, it violates the DRY design principle. Indeed, each CSC needs to be represented twice (in the text and in the MV), and therefore needs to be maintained twice. For instance, when an author updates an annotated CSC, then both versions need to be updated.

In addition, it lacks flexibility and may not satisfy the requirements of all Web sites. For instance, the value of sales tax are almost added to the value of price $CSCs$. As already discussed in Section 3.3.5, most Europe countries use VAT system (with different tax rates for each country), while Australia uses GST system with 10% as a tax rate. Up to the date of writing this section (i.e., July 22, 2010), all Microformats specifications that include price $CSCs$ such as $hListing$, $hProduct$, and $hReview$ do not

[2]More information available on: `http://microformats.org/wiki/measure`.
[3]Before it is updated, this date is not shown in this Figure

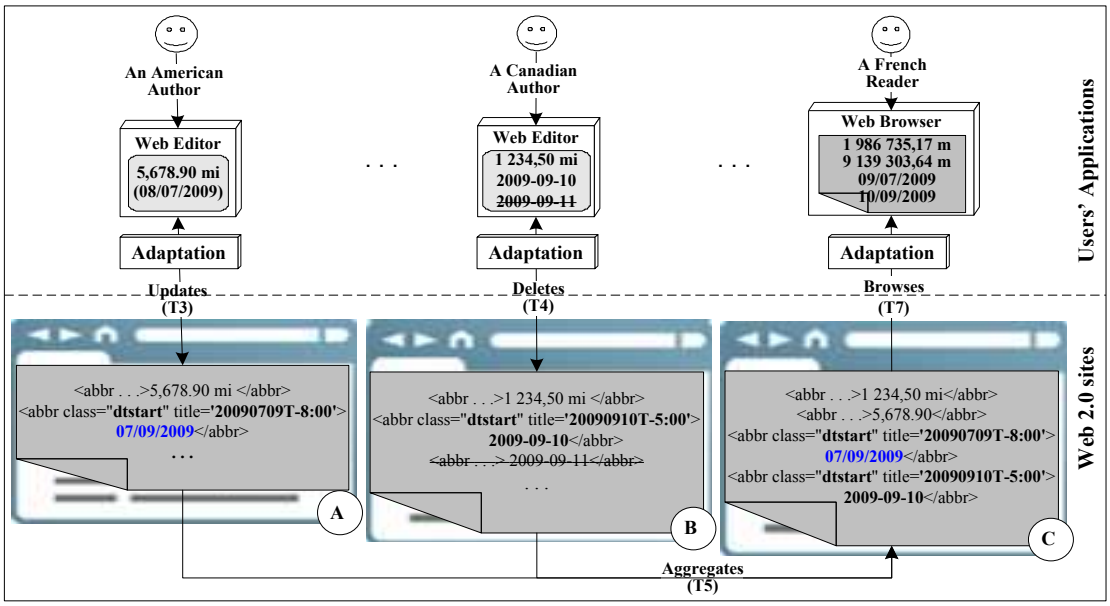

Figure 4.1: Adaptation to a Standard Local Context: Illustration Example

consider the representation of sales tax value that might be added to price *CSCs*. In contrast, introducing new Microformats specifications or extending existing ones require an extensive discussion with the Microformat community for a general adoption, as already mentioned in Section 2.3.3. Finally, each *CSC* needs to be adapted twice: one from the author's version to the *MV* version and one from the latter to the reader's version.

4.2.2 Adaptation to a Single Page Local Context

The second alternative imposes a unified local context for each Web page. In this setting, the value of each *CSC* has to be adapted from an author's context to a page's context at creation and update time. Also, its value has to be adapted from one page's context to another page's context at aggregation time, as each page has a unified context. Finally, there is a need to adapt its value to different readers' contexts at browsing time.

In practice, this alternative can be embodied using a specific representation model for each Web page (or for each Web site in general). The role of the representation model is to represent the context information and their relations with *CSCs* at conceptual level. According to this model, the context information for both a Web page and users (i.e., authors and readers) have to be specified and stored. Then, each *CSC* is adapted from one context to another by utilizing one or more conversion functions. Each function

takes this *CSC* and its suitable context information related to both a Web page and a user as parameters. Then, convert the former from one context to another (i.e., from an author's to a page's context, from a page's to another page's context, or from a page's to a reader's contexts).

Example

To understand the consequence of this alternative on Web 2.0 use cases, let us assume French, Canadian, and American local contexts are associated with pages A, B, and C involved in our example (Figure 1.1), respectively. Also, let us assume that a number of adaptation functions are utilized to adapt *CSCs* from users' to pages' contexts and vice versa, as it is shown in Figure 4.2.

In page A, Web contents created by the British author at task T1 are adapted according to the French context (i.e., the context of the page A). Then, they are adapted according to the context of the American author when he browses these contents at Task T2. Afterwards, they are adapted again to the French context after this author updates and republishes them again at task T3. In page B, there is no need to adapt *CSCs* when the Canadian author interacts with this page, as they have the same contexts (i.e., Canadian context).

Next, the contents of pages A and B are adapted to the American context when they aggregated to page C at task T5. Finally, the contents of page C are adapted to the French context when the French author browses them at task T7.

Discission

This alternative does not violate the DRY principle and does not impose a standard context for all Web sites. Moreover, Web contents in a single Web page are homogeneously represented. However, *CSCs* need to be adapted many times. These adaptations are often not necessary. For instance, assume the British author above needs to update the date content he created before. Here, this date needs to be adapted to his context, since it was adapted from his context to the page's A contexts at creation time. Also, it needs to be adapted again from his context to the page's A contexts after the last update. In addition, when the date and length contents are aggregated from pages A and B to page C, then other unnecessary adaptations are needed to adapt the aggregated contents according to the context of page C.

Finally, this alternative has to consider the inter-operability of the representation model among users applications (i.e., Web editors and browsers) and a specific Web

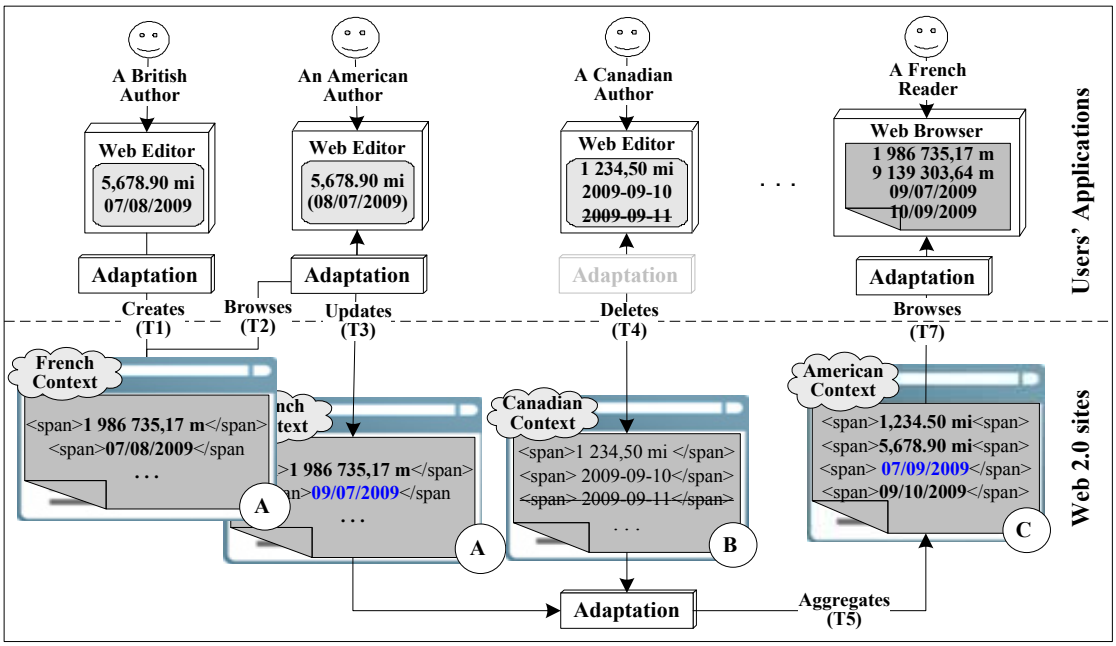

Figure 4.2: Adaptation to a single page Local Context: Illustration Example

page, and also among Web pages from different Web sites. In other words, the representation model has to rely on a common conceptualization basis in order to ensure correct adaptations from one context to others (See Section 4.3.3).

4.2.3 Annotation of *CSCs* with Authors' Local Contexts

Instead of imposing a standard or a single local context, the third alternative is to annotate *CSCs* with their authors' contexts at creation and update time, and to adapt the annotated *CSCs* to their different readers' contexts at browsing time.

In practice, this alternative also requires a representation model to represent the context information and their relations with *CSCs* at conceptual level. Based on this model, the context information for Web authors and readers have to be specified and stored. Then, the annotation process is utilized to annotate each *CSC* with a suitable author's context information. Finally, the adaptation process is utilized to adapt the annotated *CSCs* to different readers' contexts.

The main difference between this alternative and the previous one as follows. Instead of adapting *CSCs* from their authors' to a single page's context at creation/update time, it annotates each *CSC* with a suitable author's context information. Then, the annotated *CSCs* are adapted to their readers' contexts at browsing time.

Example

Figure 4.3 illustrates the annotation and adaptation of *CSCs*, mainly date *CSCs*, presented in Figure 1.1 with corresponding authors' contexts information. Here, it is assumed that the contexts of users are represented and also a number of annotation and adaptation modules are installed.

In page A, the date *CSC* created at task T1 is annotated with date format and time zone information related to the British author's context. Then, the contents of this page are adapted according to the American author's context when he browses them at task T2[4]. Afterwards, when this author updates the date *CSC* involved in this page at task T3, the annotation is also updated according to his context. In page B, no adaptation and annotation is needed when the Canadian author interact with this page, as he browses and deletes contents he created before.

Next, the contents of pages A and B together with their annotation are aggregated to page C at task T5. Finally, the contents of the latter are adapted to the French context when the French author browses them at task T7.

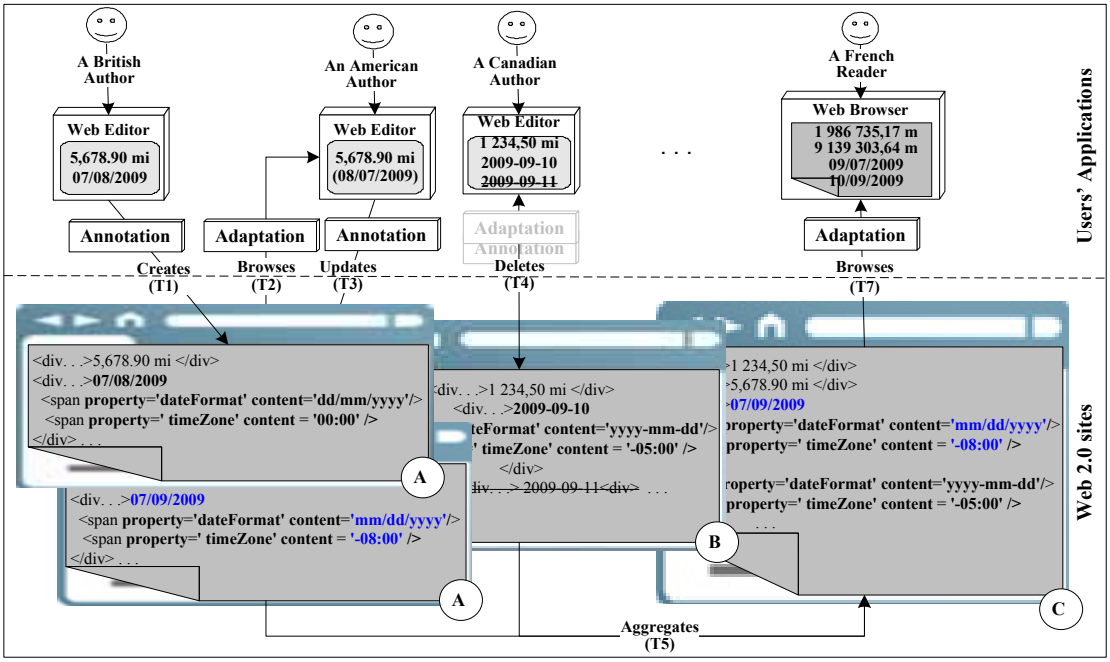

Figure 4.3: Adaptation to a Standard Local Context: Illustration Example

[4]At this task, this author is considered as reader, as already mentioned before

Discussion

This alternative does not violate the DRY principle and does not impose a standard or a single context. Furthermore, it preserves the initial Web contents as they were submitted to the Web page, which may be useful for their in-depth understanding or analysis. Also, it optimizes the number of required adaptations of $CSCs$. However, this alternative has to consider the inter-operability issue related to the representation model.

4.2.4 Design Alternatives: Conclusion

Our proposal is to adopt the third design alternative, since it is the best tradeoff with respect to the identified design criteria/principles, as summarized in Table 4.1 below.

Design Alternatives	**Standard Context**	**Single Page's Context**	**Multiple Authors' Contexts**
Context Representation Flexibility	No	Yes	Yes
DRY Compliance	No	Yes	Yes
No. of Adaptations	Twice	Many	Once
Inter-operability	Inter-operable	Require a common conceptualization	Require a common conceptualization

Table 4.1: Evaluation summary of the design alternatives.

However, several issues related to this alternative are not addressed yet, in addition to the above inter-operability issue. First, there are several annotation techniques (mainly external and internal). Each technique relies on different technologies to annotate $CSCs$ as semantic objects, and it has several consequences on Web 2.0 use cases. Second, as Web authors are usually non-expert users, annotation of $CSCs$ is still a complex process. Indeed, we should consider that authors often do not know the relations between $CSCs$ and local context information. They also do not have theoretical and technical knowledge about the annotation process. Therefore, it is difficult for authors to do it manually. On the other hand, automating this process is questionable and could leads to undesirable results, since automatic annotation faces the problem of concepts disambiguation, as it was discussed in Section 2.3.1. Therefore, Web authors should be assisted to accomplish this process. For instance, when an author needs to update a CSC created by another author, the Web editor must take care to update the annotation too (hidden from the user).

The rest of this chapter addresses the design of the representation model and its related inter-operability issue. The issues related to $CSCs$ annotations and adaptation will be discussed in Chapter 5 and Chapter 6, respectively.

4.3 Semantic Object

The adopted design alternative requires a model to enrich (annotate) each CSC with a suitable author's context information. To this end, our representation model is mainly built based on the notion of *semantic object*. Basically, the idea of semantic object lies into enriching data objects with explicit context information in the form of metadata to facilitate their automatic interpretations among heterogenous systems.

The semantic object notion has initially used in [100] to exchange data objects among heterogenous databases. Afterwards, MIX model [24] has used semantic object to automatically integrate semi-structured, complex data objects (e.g., air flight information) from heterogenous Web-based applications to an intermediary Web-based application (i.e., from online reservation systems to an intermediary travel agent). Finally, it has used by Mrissa et al. [82] to annotate data objects exchanged between Web services (i.e., message elements of WSDL[5] document) with context information related to their providers, so that this annotation facilitates automatic data mediation during Web services composition.

Following these approaches, semantic object notion is used in our approach to enrich the representation of $CSCs$ with semantic metadata in order to allow their automatic adaptation in Web 2.0. Semantic metadata consists of a set of context attributes to explicitly describe the local context information used by an author to represent a CSC. In addition, it consists of a *concept* that specifies the relation between a CSC and the real world aspect it describes (e.g., date). A semantic object $SemObj$ can be formalized as a triple as follows:

$$\boldsymbol{SemObj} = \langle S, V, C \rangle, \text{ such that}$$

- S represents a real world concept that the $SemObj$ adheres to.
- V is the physical representation (the value) of CSC.
- C specifies a minimum set of local context attributes $\{c_1, c_2, ..., c_n\}$, n $\in \mathbb{N}$. These attributes are used by an author to represent the value V of the $SemObj$. Also,

[5]WSDL refers to Web Service Description Language.

they are needed to automatically adapt the value V of the *SemObj* to other users' local contexts.

Example

As already described in Section 3.3, Web authors represent date *CSCs* using different date formats and different time zone conventions. According to the semantic object notion, this information has to enrich the representation of date *CSCs*, as context attributes C, in order to facilitate their adaptation. Figure 4.4 illustrates the representation of the date *CSC* from our scenario updated by the American author during Task T3 as a semantic object (see Figure 1.1). Here, *date* refers to the *date* concept *S*. *'07/09/2009'* represents the value V of the date *CSC*. Finally, *context* consists of two context attributes: *date-format* with the value *mm/dd/yyyy* and *time-zone* with the value *-08:00*.

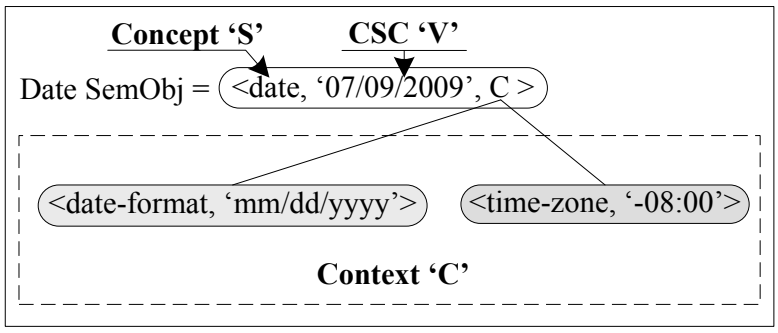

Figure 4.4: Sample of date semantic object

4.3.1 Static and Dynamic Context Attributes

In the above example, the date format and time zone attributes provide enough context information to automatically adapt the above date *SemObj* into different readers' contexts. Indeed, the date *SemObj* can be adapted to other date formats and time zones, as the latter attributes are specified explicitly.

However, the values of these context attributes are often difficult to specify. For instance, an author might face difficulty to specify the value of time zone convention, as s/he simply might not know it. Also, several countries use the daylight saving time convention to adjust the clock forward one hour near the start of the spring season and adjust it backward one hour near the start of the autumn season. In contrast, it is easier for authors to specify the name of the city related to this time zone instead. Then, the

name of the city can be used to determine the time zone convention, since the latter depends on the geographical location of this city.

Similar scenario might be happend with respect to the *date format* attribute. In practice, it is difficult for users to specify the value of date formats, even they implicitly represent date $CSCs$ according to their local date formats. In practice, the members of a community normally share the same date formats (i.e., by conventions). Hence, the value of date format attribute can be determined from the community that a user belongs to. As we will see next, a community can be identified by the natural language used by its members together with the country that they live in. The specification of the latter values (i.e., language and country values) are considered not difficult (see relations between community and community conventions in Section 4.5).

From readers' perspectives, the adaptation of $CSCs$ could also depend on context information whose values are dynamic. For example, the adaptation of a *price* content depend on the currency exchange rate between author's and readers' currencies. This rate could be changed each day or several times per day. Therefore, it is not easy or even impossible to store its value statically. However, its value can be calculated if the values of the two currencies and the time of exchange are known. Also, the adaptation of a telephone number contents depends on the value of international prefix attribute. This attribute varies based on countries of readers who browse this telephone number (see Section 3.3.3). Thus, this value can be specified at browsing time only.

To simplify the specification of their values, we use the semantic object notion to represent local context attributes C as semantic objects. Each attribute may be enriched with a set of additional context attributes in a tree structure fashion. The leaves of the context tree are $SemObj$ with *zero* context attribute[6]. Hence, the local context attributes C can be formalized as a finite set of $SemObj$ as follows:

$$\mathbf{C = \{\langle}S_1, V_1, C_1\rangle\mathbf{, \ldots, \langle}S_n, V_n, C_n\rangle\mathbf{\}},\ \mathbf{n} \in \mathbb{N}.$$

Additionally, we exploit the categorization that has been introduced in [82] and categorize context attributes into two subsets: static and dynamic. *Dynamic attributes* refer to context attributes that are often difficult to specify or difficult to store their values. Furthermore, their values can be inferred from the values of other context attributes. Consequently, each dynamic attribute is enriched with additional one or more context attributes. The values of the latter attributes are used to infer the value of this dynamic attribute. In contrast, *static attributes* refer to context attributes whose values

[6]Local context C for the leaves $SemObjs$ is an empty set.

can be easily[7] specified by users and can be statically stored. Furthermore, the values of these attributes are usually used to determine the values of other context attributes (i.e., dynamic attributes). Consequently, the values of static attributes have to be specified explicitly.

In conclusion, dynamic attributes are represented as semantic objects that have one or more context attributes. The latter could be dynamic or static attributes. Static attributes are represented as leaves semantic objects and their values have to be explicitly specified. The values of leaves semantic objects are used to determine the value(s) of dynamic attribute(s) that exist at higher level, and the value(s) of the latter determine(s) the values of the attributes at the higher level. This is recursively continued until all values in the context tree are determined.

As we will see next, the design of local context ontology complies with this categorization and provides a mean to infer the values of dynamic attributes from the values of static ones (See Section 4.5).

Example

In Figure 4.5, we refine the representation of the above date semantic object using the notion of static and dynamic attributes. As already mentioned, it is difficult to specify the date format and time zone attributes. As a consequence, we represent them as dynamic attributes and enrich each of them with a set of additional context attributes. Here, the time zone attribute is enriched with two static attributes: *city* and *country*. The values of the latter attributes is specified explicitly (i.e., 'California' and 'US') and used to determine the value of the time zone attribute (i.e., '- 08:00').

Similarly, the date format attribute is enriched with *country*, *language*, and *dateStyle*. These attributes are also represented as static attributes with 'US', 'EN', and 'short' as explicit values, respectively. Finally, these values determines the values of *dateFormat* (i.e., $mm/dd/yyyy$).

4.3.2 Usability Vs. Flexibility

In an ideal situation, inferring dynamic attributes from static ones simplifies the specification of context information, as the inferred values usually represent the intended context information that users want. In a real world situation, some users may prefer specific values other than the inferred values.

[7] By easy, we mean that users are able to specify the values of these attributes accurately and without require additional efforts (e.g., time) to specify them. In other words, we refers to the effectiveness and efficiency criteria defined in Section 1.2.2

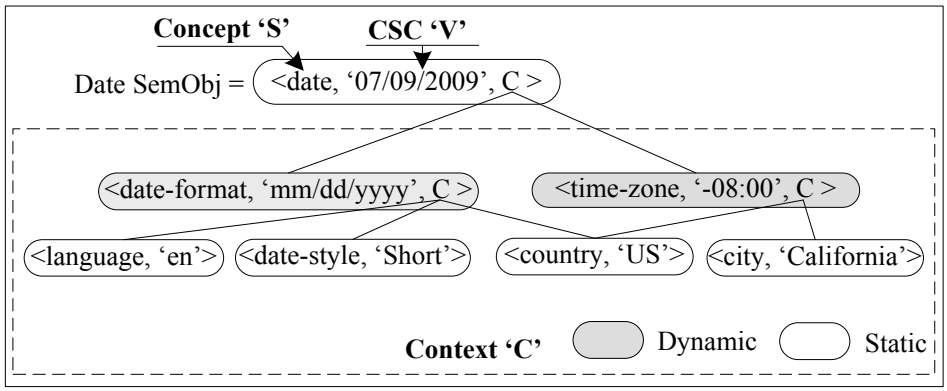

Figure 4.5: refined version of the aforementioned date semantic object

One could argue that users can simply change one or more static attribute related to the preferred dynamic attribute in order to get the value of the latter. However, the value of a static attribute is usually used to determine the values of several dynamic attributes, and vice versa. Consequently, the values of one or more dynamic attributes that a user intends to keep them are changed as long as the value(s) of static attribute(s) related to them are changed. In conclusion, there will always be a contradictive scenario between dynamic attributes that depend on the same static attribute(s).

To illustrate this, assume the American author in the above date semantic object lives in London, and he need to represent this date according to the American date format. Thus, he needs to specify the value of *country* and *city* as 'UK' and 'London' in order to infer the time zone related to the London city (i.e., '00:00'). On the other hand, he needs to specify the values of *country*, *language*, and *timeStyle* as 'US', 'EN', and 'short' in order to infer the American date format (i.e., *mm/dd/yyyy*). However, the author can only choose one country value. Therefore, one of the inferred values does not express the intended value that this author wants.

To allow users specify context information in usable and flexible manner, we utilize the following strategy. First, relying on the values of static attributes to infer the values of dynamic attributes, as already described above. Second, let authors to specify or change the values of one or more dynamic attributes as long as they want specific values. Hence, the specified values of these attributes override the inferred values in this case. For instance, the American author can specify the value of time zone attribute explicitly in the above example (i.e., '00:00').

4.3.3 Semantic Object Inter-Operability

The use of semantic object notion provides an explicit and self-contained representation of $CSCs$. However, readers' applications must agree on this representation in order to interpret and adapt these objects. This implies that authors' and readers' applications (i.e., Web editors and Web browsers) must be based on a common conceptualization.

The term *Common conceptualization* refers to an agreement and commitment by multiple applications about a domain of discourse, so that they can inter-operate in consistent manner. In contrast, these applications do not necessarily have the same experiences, theories, or prescriptions about that domain [50]. Basically, a common conceptualization is a prerequisite to foster the inter-operability of semantic objects.

From technical viewpoint, relying on ontologies for achieving a common conceptualization is considered as a reliable design strategy in Web domain. In effect, it is highly recommended as best design practices to consider existing ontologies, standards, and any other resources in the same or similar domains. There are many common ontologies available on the Web and they cover almost all domains. Concepts (vocabularies) from common ontologies can be directly reused, refined, extended, or simply used for defining mappings from new concepts to the existing ones. In addition, existing standards and resources can be indirectly exploited to design new ontologies in case there is no common ontology that covers the intended concepts or domain [91]. Then, concepts from existing common ontologies or from new ones can be used by participated applications to consistently inter-operate.

With respect to semantic objects inter-operability, three levels of common conceptualization can be identified. Each level allows users' applications to interpret a specific aspect of these objects as follows:

- **Level 1.** The real world concepts S that $SemObjs$ adhere to are common. This level allows users' applications to interpret the relations between $CSCs$ and the real world aspects they describe, as already described. However, it does not allow these applications to adapt the values of these $CSCs$ according to their readers' contexts.

- **Level 2.** The minimum set of local context attributes used to represent and interpret $CSCs$ are common (Figure 4.4 shows the representation of date CSC with a minimum set of context attributes). This levels allows readers' applications to adapt the values of these $CSCs$ as long as the values of these attributes are known.

- **Level 3.** The relations between static and dynamic context attributes are common. This allows readers' applications to adapt the value of $CSCs$ as long as the values of their corresponding static attributes are known. Indeed, readers' applications infer the values of dynamic attributes from the values of the corresponding static attributes, and then adapting their values according to readers' contexts, as already described above.

In an ideal situation, the above three levels have to be reached among users' applications to ensure a full interpretability of semantic objects. However, the third level restricts the extensibility of the context model, as the relations between static and dynamic contexts attributes have to be common. In practice, any change on these relations requires agreements and commitments by all participated users' applications. In addition, this level adds extra overhead on readers' applications, as they need to infer the value of dynamic attributes related to both authors' and readers' contexts. Note that, there is an overhead added on these applications, as they have to adapt the values of $SemObjs$.

Alternatively, we can rely on the second level above. This level ensures a consistent inter-operability among users' applications as long as they agree on concepts S and the minimum set of contexts attributes for each semantic objects. Also, it does not restrict the extensibility of the context model, since each user's application is responsible for managing the relations between static and dynamic context attributes related to his user (i.e., inferring the dynamic attributes from the static ones). However, as we will see in the next chapter, it imposes a restriction on the way context attributes are stored and also on the way they are associated with $CSCs$.

Therefore, we set a strategy, which relies on the level 2, to ensure a consistent inter-operability of semantic objects among users' applications as follows:

- Reusing a number of common ontologies available on the Web to identify mappings from the concepts S to concepts from these ontologies. The role of these mappings is to push the concepts S that $SemObjs$ adhere to them into a common conceptualization (i.e., achieving level 1 described above).

- Introducing an ontology, called local context ontology, to push the local context information used by users to represent and interpret $CSCs$ into a common conceptualization. This ontology assembles all context information into a complete conceptual model based on a set of existing local conventions standards. Also, it embodies the relations between static and dynamic context attributes.

- Evaluating several annotation alternatives. On the basis of this evaluation, the RDFa specification is utilized to associate each CSC with a concept S and its corresponding minimum set of context attributes.

The introduced context ontology is core element of this strategy. It provides vocabularies that allow both authors and readers to specify their own context information (at least static attributes). Also, it allows their applications to infer the values of dynamic attributes from the values of static ones. Users' applications can directly rely on this ontology or they can extend it according to their specific requirements. Finally, the third step ensures a common relations between $CSCs$, concepts S, and a set of context attributes C, as the RDFa specification is a W3C recommendation and therefore it is considered common in the Web domain. In the following sections, we discuss the first and the second steps in more details. The third step will be discussed in the next chapter.

4.4 Reuse of Common Ontologies

On the basis of the strategy introduced in Section 4.3.3, a number of ontologies available on the Web are investigated to be reused. The following list summarizes some of the investigated ontologies:

- *vCard* is a specification used for describing people and organizations based on RFC 2426[8] standard. W3C introduces an ontology to represent this specification as a Web standard. The RDF version of this standard is available on: `http://www.w3.org/2001/vcard-rdf/3.0#`.

  Several vocabularies from the vCard ontology can be reused to provide common concepts S for several semantic objects. For example, *tel* vocabulary can be reused as a common concept S to refer to telephone number $CSCs$. Also, tz can be reused to refer to the time zone context attribute.

- *vCalendar (vCal)* is another RFC specification used for describing events and calendar information[9]. Also, W3C introduces an ontology to represent this specification. the RDF version of this ontology is available on: `http://www.w3.org/2002/12/cal/ical#`.

[8] Available on `http://www.ietf.org/rfc/rfc2426.txt`
[9] Available on `http://www.ietf.org/rfc/rfc2445.txt`

As vCalendar is dedicated to describe events, it provides rich vocabularies related to date and time $CSCs$. For example, $dtstart$, $dtend$, $dtstamp$, $rdate$, and $lastModified$ can be reused to refer to the date and/or time $CSCs$. Also, $Vtimezone$ can be also used to refer to the time zone context attribute.

- *Dublin Core (dc)* is a set of metadata elements (vocabularies) that are used for describing resources such as electronic documents, video, text, etc. The semantics of Dublin Core metadata were established and maintained by an international, cross-disciplinary group of professionals from librarianship, computer science, text encoding, museums, and other related fields. Initially, Dublin Core Metadata Initiative (DCMI) has introduced fifteen metadata elements for describing electronic documents called *Simple Dublin Core* metadata. The RDF version of these metadata is available on: `http://purl.org/dc/elements/1.1`. Then, additional metadata vocabularies have been introduced and called Qualified Dublin Core. *Qualified Dublin Core* includes three additional elements (Audience, Provenance and RightsHolder), as well as a group of element refinements (also called qualifiers) that refine the semantics of the elements in such ways that might be useful in resource discovery[10]. The RDF version of the qualified Dublin Core metadata are available on: `http://purl.org/dc/terms/`.

  Several vocabularies from Dublin core metadata can be reused to cover the concept S. For example, *date*, *created*, *issued*, and $modified$ vocabularies can be reused as S to refer to date $CSCs$. Also, the *language* vocabulary can be reused to refer to the language context attribute.

- *The Friend Of A Friend (FOAF)* ontology provides vocabularies that are used to mark up people and their relationships with others such as personal information, interests, mail box, etc. W3C introduces an RDF-based version of FOAF as a Web standard. This standard is available on `http://xmlns.com/foaf/0.1/`.

- *W3C Units ontology (un)* is introduced by W3C for representing units of measure as members of class unit. The purpose of unit ontology is to provide a set of unit instances as common vocabularies used to measure physical quantities. Each unit instance is represented using the common measure code of this unit. For example, the instance of *meter* is represented using m measure code. Therefore, we can reuse the unit class as a common vocabulary for our *unit* context attribute and

[10] More information available on `http://dublincore.org/`

the unit instances as values V for this attribute. The RDF version of the units ontology available on: `http://www.w3.org/2007/ont/unit#`.

- *Measurement Units Ontology (muo).* is another ontology dedicated for representing units of measures, unit prefixes, physical quantities, and the relations between them. It has three main classes: $UnitOfMeasurement$, $PhysicalQuality$, and $Prefix$. It also has a $MeasuredIn$ property that defines a relation between the former two classes (i.e., $PhysicaQuality$: $MeasuredIn$: $UnitOfMeasurement$). In addition, this ontology provides a number of instances for the above classes. The objective of the latter is to provide vocabularies used to refer to the most common units of measure (e.g., meter and gram), physical quantity (e.g., length and weight), and measure prefixes (e.g., kilo and mega). Hence, we can reuse this ontology to refer to the physical quantity $CSCs$ and context attributes related to them. The OWL version of this ontology available on: `http://purl.oclc.org/NET/muo/muo-vocab.owl`.

- *Good Relations (gr)* is a lightweight common ontology that provides a set of vocabularies to allow users (i.e., sellers, manufacturers, and online shop operators) to express the meaning of their offers made on the Web in a machine readable way. It provides rich vocabularies related to price $CSCs$ and its context attributes. For example, the concept $PriceSpecification$ is used to specify the prices of offers (i.e., products or services). Thus, the latter can be reused to refer to price $CSCs$. Also, the $PriceSpecification$ has several attributes such as $hasCaurrency$, $hasCurrencyValue$, and $valueAddedTaxIncluded$. These attributes can be reused as common vocabularies to refer to context attributes. The RDF version of this ontology available on: `http://purl.org/goodrelations/v1`.

- *DBpedia Ontology* is cross-domain ontology, which has been manually created based on the most commonly used infoboxes within Wikipedia. Infoboxes are templates that represent structured data contained in many Wikipedia articles. Indeed, the Infoboxes are classified into hierarchy consists of 170 ontology classes and linked together using 720 ontology properties. Therefore, this ontology provides huge number of vocabularies. In addition, there are a huge numbers of instances created for each of the DBpedia class. For example, *country* and *city* are two classes covered by the DBpedia ontology. The instances of these classes are almost consists of all world's countries and cities. These instances can be reused as domains for these concepts.

These ontologies are considered as a good starting point towards common conceptualizations. They provide rich vocabularies that almost cover all concept S and context attributes C for the above semantic objects. Also, they are adopted by an increasing numbers of users. Finally, the instances created by these ontologies (e.g., the instances of units and the instances of countries and cities) can be reused as common domains for several context attributes.

However, the relations between concepts in these ontologies are specified in different ways, and oriented to the application domains that already designed for. More specifically, these relations do not specify the underlying relations between $CSCs$ and context attributes, such that the latter can be adapted to different readers' contexts. For instance, ontologies that provide concepts to refer to price CSC are almost do not specify the price format attribute.

Consequently, we rely on these ontologies to identify a direct mapping between the concepts S that $SemObjs$ adhere to and concepts from these common ontologies. Table 4.2 presents these mappings. Note that the URIs of these concepts are presented using a CURIE syntax[11]. In addition, these ontologies are indirectly exploited in the design of the local context ontology described in the next section.

4.5 Local Context Ontology (LCO)

In the above proposed strategy, we identify the following two roles of the LCO. First, assembling local context information into a complete conceptual model and pushing them into a common conceptualization. Second, embodying the relations between static and dynamic context attributes.

This section details the design of the LCO. We start by discussing our interpretation of the local context definition. Then, we reflect this interpretation into a set of concepts and relations between them in the LCO. In addition, we backing our design by a set of initiatives and standards that consider one or more aspects related to local context.

[11]CURIE (or compact URI) is abbreviated syntax introduced by W3C for expressing URI of a vocabulary as a prefix:reference. The prefix is a mapping from ontology URI to such abbreviation, and the reference represents the vocabulary itself. For instance, $vCard : dtstart$ is a CURIE syntax, such that $vCard$ is a compact mapping from the URI: `http://www.w3.org/2001/vcard-rdf/3.0#` to $vCard$ prefix and $dtstart$ is the vocabulary itself. More information about CURIE available on: `http://www.w3.org/TR/curie/`.

Concepts S	**Concepts From Common Ontologies**	**Description**
Date/Time	vCal:dtstart, vCal:dtend, vCal:dtstamp, vCal:rdate dc:date, dc:created, dc:issued, dc:modified	A time duration during a particular day, or a day of a month or a year represented within a calendar system
Price	gr:PriceSpecification	A numerical monetary value, in a specific currency, assigned to a good or a service to exchange them between sellers and consumers
Physical Quantity	un:quantity mou:PhysicalQuantity	A representation of a quantifiable entity such as length, weight, and temperature. Each entity is called quantity dimension and measured using a combination of a number and (prefixed) measure unit
Telephone	vCard:tel	A unique sequence of decimal digits used to identify a telephone line in a telephone network and to make a telephone calls among telephone lines inside a country or a cross countries

Table 4.2: Mapping between CSCs and concepts from common ontologies

4.5.1 Local Context: Interpretation and Main Concepts

The local context is defined in Section 1.2 as a common knowledge and local conventions shared between local community members. In effect, community members follow several conventions that are used/issued in a country who they live in such as currency and sales tax system. Also, one country has many cities or states. These cities/states might be located in different geographical zones, and hence different local times are used (i.e., time zones). In addition. one country may have one or more communities (e.g., French and Dutch speaking communities in Belgium). Each community usually uses a common natural language and a set of conventions related to that community such as the notational writing formats.

Consequently, we design *LCO* around *local-context* as a key concept. Next, the latter is divided into two sub-concepts: *country-convention* and *community-convention*. In other words, the latter sub-concepts is *part-of* the *local-context* concept. The role of this division is to provide an abstraction of the local context conventions and relate them to their origins (i.e., country and community). These relations are represented as follows. We initially add *country* and *community* concepts and *is-used-in* property to

the *LCO*. Then, each type of convention is related to its origin using *is-used-in* property. This means that a country convention is used in a country and a community conventions is used in a community. Figure 4.6 illustrates the main concepts of the *LCO* ontology.

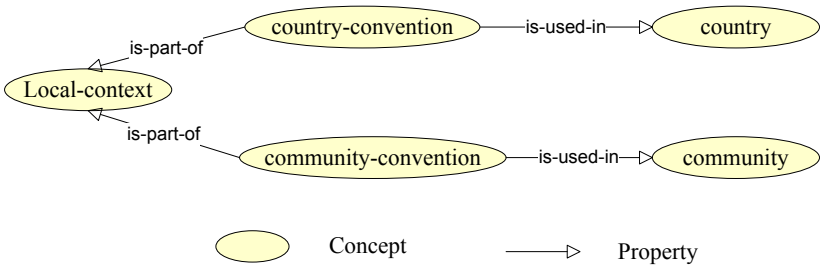

Figure 4.6: Local Context Ontology: main concepts

The division of local context into country and community conventions and relating the latter into their origins is derived from the *locale model*. This model is provided in the Unicode Locale Data Markup Language (LDML[12]). LDML defines *locale* as an identifier used to specify a set of users preferences and to determine the presentation of locale-dependant data in computer interfaces. This identifier is primarily consists of language code taken from ISO 639, country code taken from the ISO 3166, and other variants such as currency code taken from the ISO 4217. Practically, Locale and locale-dependant data are represented as XML documents in LDML langauge.

In this sense, our local context definition is very closed to the locale definition. However, since XML focus on defining the syntax and structure of data, the semantics of locale and relations with locale-dependant data remains implicit in XML documents. Applications that exchange these data must agree on their semantics and hard-code the relations between them in advance [23]. Furthermore, LDML does not cover all *CSCs* and their related context information as defined in this work. For example, physical quantities, telephone numbers and their related context information are not covered in the LDML.

4.5.2 Country Convention

A country convention is an abstract concept used to relate a number of conventions to their originated country, as already mentioned above. In practice, one country convention could be used in one or more country. For instance, Euro currency is a country convention used in many Europe countries. The inverse case could exceptionally happen. For

[12]More details available on: `http://unicode.org/reports/tr35`, last visit 23/03/2010.

instance, it is an exception for one country to use different currency or different sales tax system[13].

In *LCO*, we simplify our design by considering the normal case. This means that one country convention could be used in one or more country, but one country use one country convention. The intention of this decision is to be able to determine the *normal* country conventions for each country. This provides a way to embody the notion of *static and dynamic* context attributes described above[14]. It is worth noting that this design decision is also derived from the locale model provided in the LDML language.

Indeed, we rely on the two letters country codes listed in the ISO 3166[15] to specify the names of countries. Also, we introduce a new property between *country-convention* and *country* concepts called *determine*. This relation is used to specify that a country determines its normal conventions. For instance, Belgium country represented as *be* determines its normal currency convention (i.e., *Euro*). Next, the *country-convention* concept is specialized into a number of concrete conventions. These includes *sales-tax-system, measure-system, tel-numbering-plan, currency, and time-zone*. Figure 4.7 illustrates the addition of *determine* property and the number of the concrete country conventions to the *LCO*.

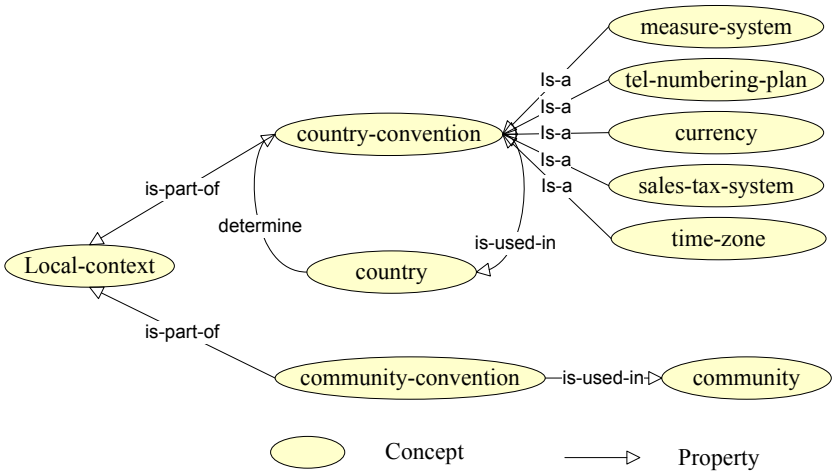

Figure 4.7: *LCO*: Addition of *determine* property and specialization of *country-conventions*.

In practice, each of these conventions is very broad and can be viewed from many perspectives. Here, we view them from local context perspective, and hence we focus on specific aspects that are used by users to represent and interpret *CSCs*. In the

[13]As an example, Palestinian authority uses two currencies: Jordanian dinar and Israeli shekel.
[14]Country is considered as a static context attribute and country convention as a dynamic attribute.
[15]The ISO 3166 standard defines two or three letters codes for representing the names of countries.

following, we define the essential nature of these conventions and discuss the intended aspects related to this work. Figure 4.8 illustrates the complete view of these country conventions.

Measure System

As already mentioned in Section 3.3.4, a measure system refers to a number of agreed measure units and unit prefixes. Each measure unit is used as a reference for measurement of physical quantities of the same type called *quantity dimension*, and also might be combined with a prefix unit to represent a multiple or a fraction of this unit. For instance, the *meter* is an agreed measure unit for the *length* quantity dimension in the *international system of units, or SI*. This represents a definite magnitude of length. Also, the *kilo* is an SI prefix that can be combined with the meter unit to denote that the meter unit is multiplied by 1000.

In *LCO*, we focus on the agreed quantity dimensions, measure units, and measure prefixes of local measure systems. Hence, we add *measure-unit* and *measure-prefix* concepts to the *LCO* and use the *part-of* property to specify them as parts of the *measure-system* concept. Finally, we add *quantity-dimension* concept and *measured-in* property. The latter is used to specify that a quantity dimension is measured using a (prefixed) measure unit involved in a measurement system.

The domain of these concepts are all agreed quantity dimensions, measure units, and measure prefixes used in the corresponding measure system. For example, SI (or Metric), Imperial, and US measure systems are instances of the *measure-system* concept. Also, meter and kilo are parts of SI measure system and represent instances of the agreed measure units and measure prefixes, respectively. Finally, meter, with or without kilo prefix, are used to measure length dimension.

Indeed, users (have to) select the measure prefix that they prefer from the set of measure prefixes corresponding to measure system they use. As measure prefixes represent multiples or a fractions of measure unit, each prefix has a scale factor. The latter is relatively easier to specify than the corresponding measure prefix. So, we add *scale-factor* concept and *has* property to *LCO*, and use the latter to relate each prefix with its own scale factor (See Figure 4.8).

Note that, the property *determine* that specified between country and country-convention concepts is propagated to these concepts, since they are parts of the *measure-system*, and the latter is a country convention. Therefore, one country determines its corresponding measure units and measure prefixes.

Country Calling code, National, and International Prefixes

Several country conventions are used to regulate the dialing of telephone numbers from inside a country and among countries, as already discussed in Section 3.3.3. These conventions are usually managed by the the International Telecommunication Union (ITU) using a plan called *International numbering plan* or E.164.

Technically, E.164 identifies the structure and the functionality of telephone numbers. Also, it identifies a number of interfaces to dial telephone numbers inside a country (i.e., national prefixes) and between countries (i.e., country calling codes and International Prefixes). Indeed, This plan identifies a unique country calling code for each country and recommends '0' and '00' as standard national and international prefixes. However, many countries still use different prefixes such as USA and Canada.

To represent these conventions in *LCO*, we add *national-prefix*, *country-calling-code*, and *international-prefix* concepts to *LCO*. Then, we use the *part-of* property to specify these concepts as parts of the tel-numbering-plan concept. The domain of these concepts are all national prefixes, country calling codes, and international prefixes used in countries (See Figure 4.8).

Note that, the property *determine* that specified between country and country-convention concepts is also propagated to these concepts, since they are parts of tel-numbering-plan, and the latter is a country convention. Therefore, one country determines its corresponding prefixes and calling code. For instance, Belgium determines 0 as national prefix, 32 as country calling code, and 00 as international prefix.

Currency

As each country determines its currency and one currency might be used in multiple countries, this convention is represented in straightforward manner in the *LCO* as follows. The concept *currency* is specified as a country convention. In addition, currency codes identified in the ISO 4217 are utilized as a domain of this convention. For instance, USD and EUR are the ISO 4217 codes for U.S. Dollar and Europe Euro.

Time Zone

Time zone is a specific type of country convention that is related to geographical location of a city or a state located in a country. By convention, the globe is divided into a number of regions, each of which has a uniform standard time, referred to as *local time*. The local time of the Greenwich region is specified as a reference time zone, and it is called the Greenwich Mean Time (GMT) or Coordinated Universal Time (UTC). Accordingly,

the local times of other regions are computed as offsets from UTC. In this work, *time zone* refers to the *local time* used in a specific region.

As already mentioned, cities or states that are part of one country could be related to different time zones, and thus could have different time zone convention. Hence, the design assumption that says a country determines its conventions can not be applied here, since the time zone convention could be related to a part of this country. In this case, we can use the *determine* property to specify that a city/state determines its time zone convention. In conclusion, the time zone convention is determined by city and country, since each city is a part of a country.

Accordingly, we design this situation as follows. We add *city* and *time-zone* as new concepts. Next, we use the *part-of* property to specify the relation between *country* and *city*, and the *detremine* property to specify the relation between *city* and *time-zone*. We also use the *is-used-in* property to consider that one time zone is used in many cities (See Figure 4.8).

Sales Tax System

Prices of goods or services might involve sales taxes. In practice, several sales tax systems and several related conventions are used in different countries. These are mainly related to whether the sales tax is included or excluded, the type of the utilized sales tax system, and the sales tax rate (see Section 3.3.5).

In *LCO*, these conventions are designed as follows. the concepts *tax-included, tax-type, and tax-rate* are added to *LCO*. Next, the property *part-of* is used to relate them to the *sales-tax-system* concept, as it is shown in Figure 4.8. As the latter concept is a country convention, the value of a country can be used to determine these conventions.

4.5.3 Community Conventions.

Like country convention, we simplify our design by considering that one community convention could be used in one or more communities, and one community use one community convention. Though, the following fundamental question rises here: what is the information that can be used to specify a community, such that users' applications can interpret the latter and determine its conventions? With respect to country conventions, a country can be specified easily by relying on the well-known, unified country codes (i.e, The ISO 3166 two letters codes). However, there is no names, codes, or other information that are common or can be used to specify communities, even between members of such a community.

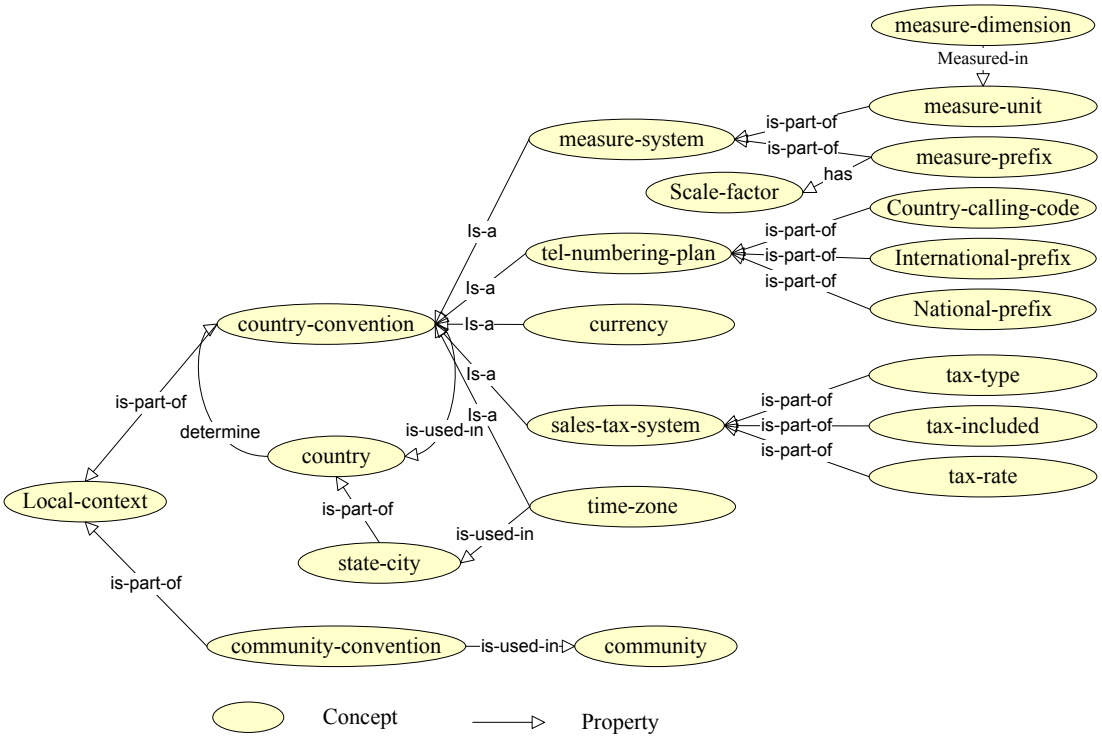

Figure 4.8: *LCO*: Complete view of the country conventions.

As community is part of a country, we can rely on the latter together with another information to work around this issue. The best candidate information is the language convention that is used in a community. In effect, a community language can be easily specified by relying on the standard language codes[16]. In conclusion, the combination of a country and a language can be used as a community identifer. This identifier is then used to determine the other community conventions, based on the above design assumption.

In *LCO*, this is designed as follows. We use the property *part-of* to specify that one or more *community* is *part-of* a *country*. Next, we specialize the *community-convention* concept into a *language* and *writing-format* concepts. Writing format considers the ways *CSCs* are formatted within a community, as it is discussed in Section 3.3. Therefore, it is specialized into the following four sub-concepts: quantity-format, phone-format, price-format, and datetime-format.

Figure 4.9 illustrates the extension of the *LCO* with concepts and relations related to community conventions. It is worth noting that each community has many conventions

[16]The ISO 639 standard defines two letters codes for representing the names of languages.

in addition to the language and the writing formats. Here, we only focus on these two conventions as they are used to represent and interpret the $CSCs$.

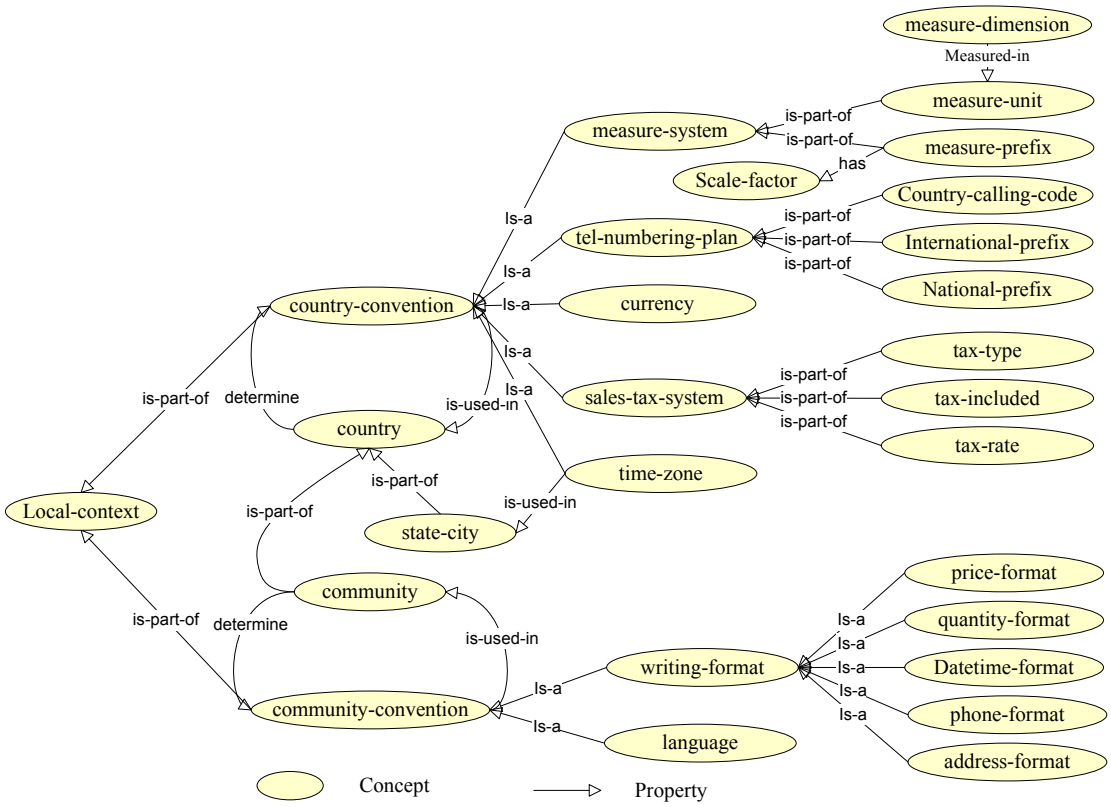

Figure 4.9: *LCO*: An extension with concepts and relations related to community conventions.

4.5.4 *LCO* Implementation

Having detailed the design of the *LCO*, the protege 4.0™ontology modeling editor[17] is utilized to implement it. Technically, each of the above concepts is represented as *owl:Class*, and each property is represented as *owl:ObjectProperty*. Also, the design assumptions such as the cardinality between the conventions and their origins are represented using the *owl:Restriction* class[18]. Figure 4.10 shows an excerpt of the *LCO* implementation.

In addition, the domain of each concept have to be specified. For instance, the domain of the concept *country* involves all countries. As already described in Section 4.4, there

[17] `http://protege.stanford.edu/`
[18] OWL language is described in more details on: `http://www.w3.org/TR/owl-ref/`

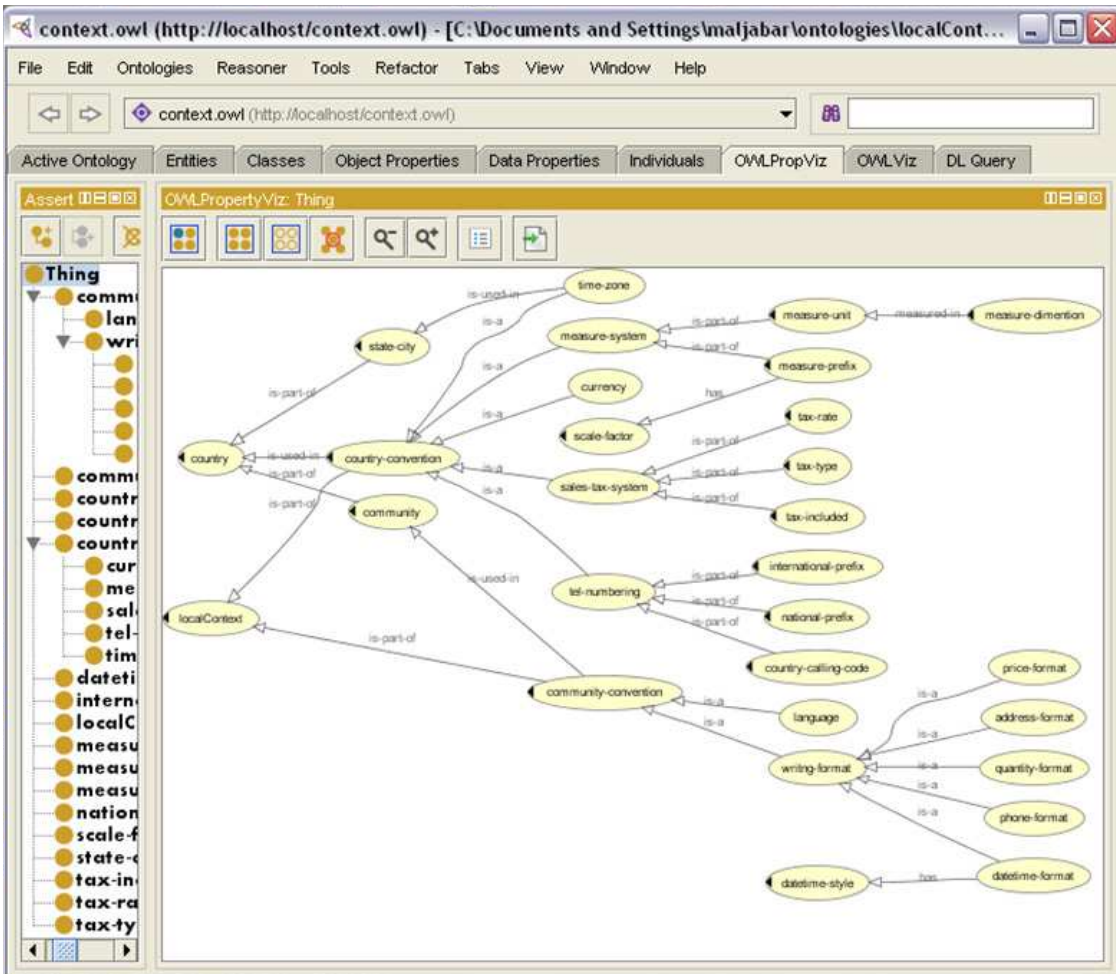

Figure 4.10: An excerpt of the *LCO* implementation using protege™(OWLProVIZ plug-in view)

are several ontologies/standards available on the Web and they can be reused to specify the domains of the *LCO* concepts. For instance, we can rely on the ISO 4217 and the ISO 639 to specify the domains of country and language concepts, respectively. Also, we can import the domains of date format and phone format from the LDML language and from the DBPedia, respectively. As an example, we create several instances of *LCO* concepts as OWL individuals.

4.6 Typical Representation of *CSCs*

On the basis of the semantic object notion and *LCO*, this section introduces a typical representation of the *CSCs* discussed in Section 3.3 as semantic objects. Like the repre-

sentation of the date/time $CSCs$ (see Section 4.3), the minimum set of context attributes that are necessary to enrich each type of $CSCs$ are classified into static and dynamic attributes. Afterwards, we illustrate how the relations between local conventions (i.e., country and community conventions) and their origins are utilized to determine the values of dynamic attributes.

4.6.1 Physical Quantity $SemObjs$

Web users represent each kind of physical quantities (i.e., quantity dimensions) using different measure units and different formats. Also, measure units might be prefixed with different unit prefixes (see Section 3.3.4). Hence, physical quantity $CSCs$ have to be enriched with these information as a minimum set of context attributes to facilitate their adaptations.

Users can easily specify measure units used to measure common or basic quantity dimensions. However, some units are derived from other measure units. For example, the unit used to measure the *energy* dimension in SI system is called *Joule*. This unit is derived by multiplying *Newton* unit by *Meter* unit. Moreover, *Newton* is derived from other units. Therefore, users could face difficulties to track and specify these units.

In addition, users could face difficulties to specify the formats used to write these quantities. Also, it is not easy for them to specify the measure prefix (if any). Therefore, these attributes are classified into static and dynamic attributes, whereas the values of dynamic ones are determined based on the LCO design as follows:

- Quantity dimension is represented as a static context attribute. So, users have to specify its value.

- Measure unit is represented as a dynamic context attribute. Also, it is enriched with *quantity dimension* and *country* as context attributes. Indeed, the value of measure unit can be determined if the quantity dimension to be measured and the measure system used are known. The measure system can be determined from the value of *country*, as it is designed as a country convention.

- Measure prefix is also represented as a dynamic attribute, and it is enriched with *scale factor* and *country* as context attributes to determine its value. The value of *country* is used to determine the set of measure prefixes corresponding to a measure system used in this country. Then, the value of scale factor determines the corresponding measure prefix from the determined set.

- Like the price format, the values of quantity format can be determined from the values of *country* and *language*. So, it is represented as a dynamic attribute, and it is enriched by the latter static attributes.

Example

Figure 4.11 illustrates the representation of the length CSC created by the British author at task T1 in our scenario as a physical quantity semantic object (See Figure 1.1). The *physical-quantity* refers to the concept S that this object adheres to. *'5,678.90'* represents the length value V, and context C consists of four context attributes corresponding to this object. Some of these attributes are also enriched with context attributes.

Here, the user has to specify the values of quantity dimension and scale factor attributes (i.e., 'length' and '1', respectively). In addition, it is assumed that the values of country and language are also specified (i.e., 'GB' and 'en', respectively). Then, the values of the latter attributes are used to determine the value of quantity format attributes (i.e., '#,##0.00 (P)U[19]'). Also, the values of country and quantity dimension determine the value of measure unit (i.e., 'mi'). Finally, the values of country and scale factor are used to determine the value of measure prefix[20]

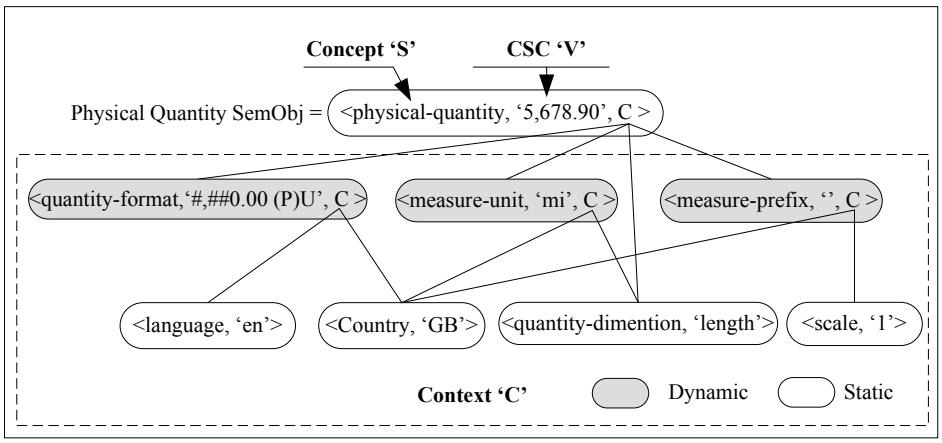

Figure 4.11: Typical representation of $CSCs$: Physical quantity semantic object sample

[19]In this format, P refers to measure prefix value and U refers to measure unit value.
[20]Here, there is no measure prefix used, as the value of scale factor is 1.

4.6.2 Price $SemObjs$

Web users represent price $CSCs$ using different currencies and different price formats. Also, the sales taxes could be included or not. If so, different sales tax types and/or rates are used (see Section 3.3.5). Hence, price $CSCs$ have to be enriched with these information as a minimum set of context attributes in order to facilitate their adaptations to their readers' local contexts.

From usability perspective, it is not easy for users to specify the values of price formats, sales tax types, and sales tax rates. So, we represent them as dynamic context attributes. On the basis of the LCO design, the values of these attributes can be determined mainly by the values of *country* and *language* attributes as follows:

- The value of a price format can be determined from the values of a *country* and *language* attributes. Indeed, the latter attributes are used to identify the community that a user belongs to. Then, the community is used to determine the price format value, as the latter is represented as a community convention and the community determines its conventions. Hence, price format attribute is enriched with *country* and *language* as static context attributes.
- As long as the sales tax is included, the values of sales tax type and sales tax rate can be determined from the value of *country*. This is because the sales tax system is designed as a country convention and the sales tax type and rate are parts of the latter. Therefore, these two attributes are enriched with *country* and *tax-included* as static context attributes.
- Users can easily specify the value of *currency* attribute. Also, the value of *country* attribute can be used to determine the latter. Thus, we decide to represent it as a dynamic attribute, and to enrich it with a *country* attribute. Users can rely on the determined values or choose their preferred values.

Example

Figure 4.12 presents a price semantic object represented according to local context of a French user. The *price* concept refers to the concept S that this object adheres to. *'5 678,90'* represents the price value V, and the context C consists of four context attributes corresponding to this object. Each of which is enriched with one or more attributes.

Here, it is assumed that the values of the *country*, *language*, and *tax-included* static attributes are specified by this user (i.e., 'fr', 'FR', and 'True', respectively). Then, the

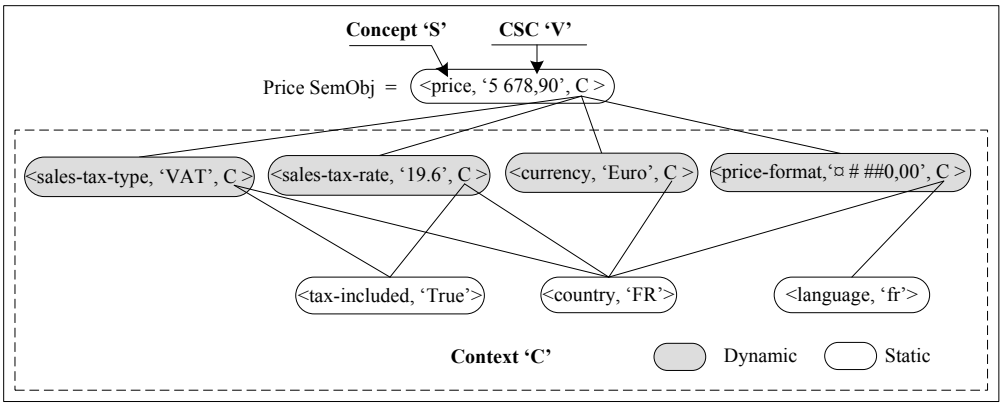

Figure 4.12: Typical representation of $CSCs$: Price semantic object sample

value of *country* and *tax-included* are used to determine the values of the *sales-tax-type* and *sales-tax-rate* (i.e., 'VAT' and '19.6', respectively). Also, the value of *country* is used to determine the value of the *currency* (i.e., 'Euro'). Finally, the values of *country* and *language* are used to determine the value of the *price-format* (i.e., '# ##0,00 ¤'[21]).

4.6.3 Telephone Number $SemObjs$

A telephone number has to be complemented with a national call prefix to make a telephone call over a local telephone network (i.e., inside a country). In contrast, the national call prefix has to be omitted and a telephone number has to be complemented with other two prefixes to make a telephone call across countries: a country calling code corresponding to destination telephone and an international call prefix corresponding to source telephone. Finally, Web users represent telephone numbers using different formats (See Section 3.3.3). To facilitate the adaptations of telephone number $CSCs$, these prefixes have to be represented separately. Next, a telephone number has to be enriched with this information as follows:

- The national call prefix and the country calling code are represented as dynamic attributes, and they are enriched with a country attribute. Indeed, as they are part of the telephone numbering plan, their values can be determined from the value of a country.

[21] In '# ##0,00 ¤', '¤' refers to the general currency symbol and it is replaced by currency symbol used (i.e., €). Also, the # refers to a digit and it is replaced by the corresponding digit value from the currency value or nothing if the value of this digit is empty in the currency value. Finally, 0 refers to a digit and it is replaced by the corresponding digit value from the currency value or 0 if the value of this is empty.

- As the international call prefix relates to the source telephone (related to reader's local context), then its value can be determined by the value of a country corresponding to a reader. This value can be specified at browsing time only, as already mentioned before. Therefore, it is represented as a context attribute with fixed value '+'. This value is determined by the value of a reader's country at browsing time.

- The phone format is represented as a dynamic attribute, and enriched with country and langauge attributes, like other formats.

Example

Figure 4.13 presents a phone semantic object represented according to local context of a British user. The *phone* refers to the concept S and *'123 4567 890'* represents the phone value V. Finally, context C consists of calling prefixes and phone format as context attributes. The international prefix has a fix value (i.e., '+'). This value is replaced by the value of international prefix corresponding to reader's country at browsing time. For instance. it is replaced by '00' when a French reader browse this phone. The values of national prefix and country calling code (i.e., '0' and '44', respectively) are determined by the value of country (i.e., 'GB'). Finally, the value of phone format is determined by the values of country and language[22].

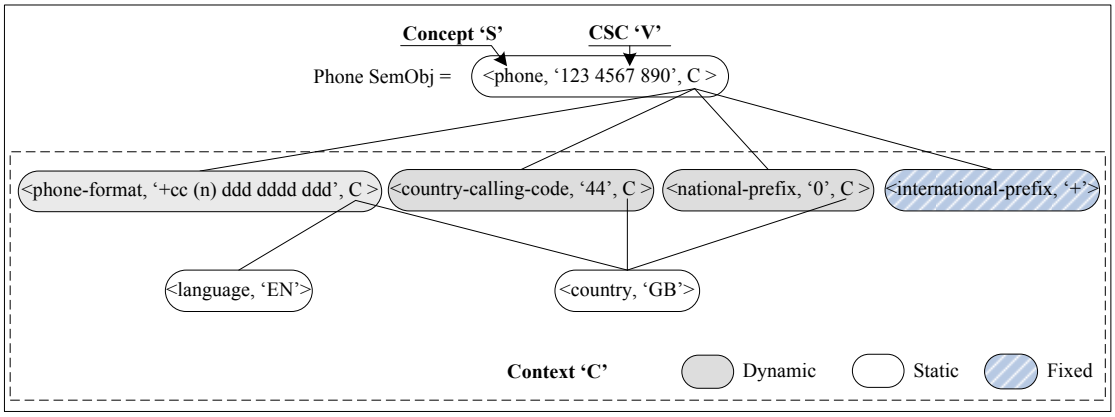

Figure 4.13: Typical representation of $CSCs$: Phone semantic object sample

[22]In this format, *cc* refers to country calling code, and *n* refers to national prefix. Also, the groups of *d* refer to digits of phone number.

4.7 Semantic Context-Aware Architecture

This section describes our vision concerning the architectural design that is intended to support our approach. Our design focus on the following three aspects. First, the main components that are necessary to accomplish the annotation and adaptation processes. Second, a high-level description concerning the interaction of Web users with these components. Third, an illustration of how our approach work seamlessly with the existing Web technology stacks.

4.7.1 Architecture Description

Figure 4.14 below depicts our proposed architecture. This architecture encompasses three layers: concept description, Extended users' applications, and Web 2.0 sites.

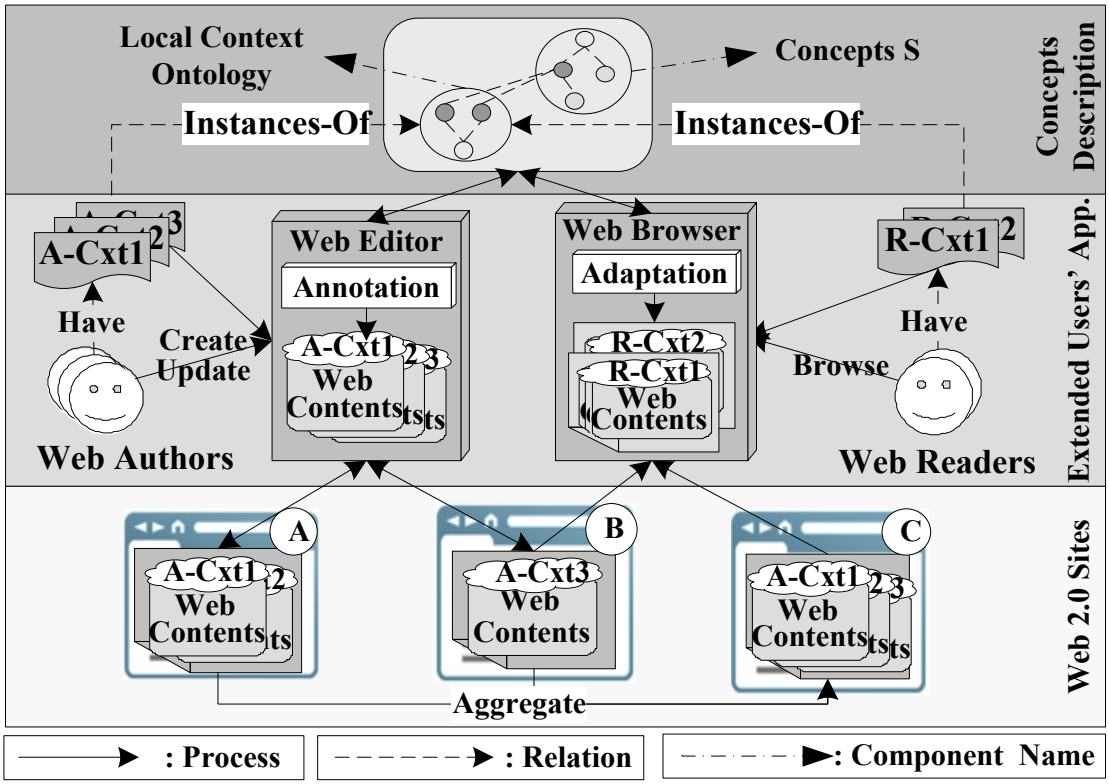

Figure 4.14: A general overview of the proposed architecture

The *concept description* layer presents the LCO, the concepts S, and the relations between them at conceptual level. These provide the sematic information that allow users' applications to interpret $CSCs$. As already mentioned, users' applications can di-

rectly rely on this information or they can extends them according to their requirements. However, they have to agree on the level 2 of common conceptualization discussed above to ensure a consistent semantic objects inter-operability.

Web 2.0 sites provide spaces for authors to upload different types of Web contents on the Web. Also, these sites could provide one or more Web 2.0 means for publishing and managing these contents (i.e., create, update, aggregate, and delete). Section 3.2 surveys several types of means that allow authors to do this. On the readers' side, the contents of these sites are browsed using traditional Web browsers such as Microsoft™ Internet Explorer and Mozilla™ FireFox.

In the *extended users' applications*, an annotation engine and an adaptation engine are proposed as extensions for both authors' and readers' applications (i.e., traditional Web editors and Web browsers), respectively. These two engines are the central constructs of this architecture, in addition to concept description layer. They provide the main components that are necessary to accomplish the annotation and adaptation processes.

As we will see in next chapter, the annotation engine is used to assist authors for annotating $CSCs$ with their context. Web authors need to specify their context (i.e., A-Cxts).Also, they need to specify the type of $CSCs$ to be annotated when they create or update them. Then, the annotation engine interactively annotates the specified CSC with suitable context attributes extracted from the corresponding A-Cxt. Finally, each annotated CSC is published to a Web 2.0 site as a semantic object $\langle S, V, C \rangle$.

On the readers' side, the adaptation engine also allows Web readers to specify their contexts (i.e., R-Cxts). Then, it adapts each semantic object from authors' to a reader's contexts. The output is semantically equivalent to the original annotated CSC, but it is represented according to a reader's context. The issues related to Web adaptation will be discussed in Chapter 6.

4.7.2 Architecture Features

The main idea of the aforementioned architecture lies into the extension of traditional authors' and readers' applications with Web annotation and Web adaptation engines, respectively. Additionally, the use of the concept description layer for enriching $CSCs$ with semantic metadata and adapt the latter according to readers' local contexts. This implies several features, which characterize our architecture. Firstly, the annotation and adaptation processes are performed at authoring and browsing times, respectively. This allows authors to annotate $CSCs$ with their local contexts at times they create

and/or update these contents. Similarly, annotated *CSCs* are adapted at times they are browsed. In addition, this optimize the number of adaptations that are performed on a *CSC* during its life cycle, as we mentioned before. Secondly, the extension of readers' applications with adaptation engines provide a way to adapt Web pages' contents that are related to a Web site, aggregated from many sites to another Web site, or aggregated from many Web sites by readers' applications (i.e., support all Web 2.0 use cases). Finally, the proposed architecture does not provoke side effect when annotated Web pages are presented on classical reader's application (i.e., not extended with our adaptation engine), or when non-annotated Web pages are presented on extended reader's application.

Apart from the above features, our proposed architecture can be reused in other semantic Web and/or Web 2.0 approaches for several purposes. Firstly, the annotation engine can be reused for enriching Web contents with different types of semantic metadata for the purpose of authoring *semantic Web contents*. For instance, Web 2.0 sites such wikis and social networking systems can rely on the annotation engine for enriching their content with semantic metadata. In this context, the semantic description layer has to be extended with additional concepts. These concepts can be taken from domain ontologies or from open data sources such as linked data cloud. Accordingly, the annotation engine also has to be redesigned and then extend Web editors that are used by these systems. Additionally, Web 2.0 recommender systems can rely on annotation engine to enable users for enriching their contents with semantic tags taken from a predefined taxonomy or from user-defined categories [**?**].

Secondly, the adaptation engine can be reused for performing different types of adaptation on semantic Web contents (i.e., annotated contents) at readers' side. For instance, Web 2.0 recommender systems can rely on adaptation engine for classifying and recommending Web contents based on the added semantic tags together with reader's context. Additionally, mush-up systems can reuse semantic metadata, which enrich Web contents, for aggregating similar types of contents or related contents. Afterwards, relying on adaptation engine for adapting the aggregated contents according to readers' contexts. It is worth noting that readers' contexts might be extended with additional context information such as readers' preferences or their devices capacities.

To sum up, the components of our architecture provide means that are necessary for authoring semantic Web contents at the same times users create/update these contents. We advocate that this is the first step that pave the road to realize the semantic Web vision. Also, our architecture illustrates how semantic Web contents are exploited by automatically adapt these contents according to their readers' local contexts.

Chapter 5

Web Annotation

5.1 Introduction

The term *Web annotation* is identified in Section 1.3.1 as a process of annotating $CSCs$ with authors' local context information in order to address the *incompleteness* of their representation. Also, Section 4.2 concludes that the annotation of $CSCs$ with their authors' local contexts at creation and update time is the best design alternative with respect to the Web 2.0 use cases. Accordingly, the semantic object notion is utilized as a formalization mean to enrich each CSC with a concept S and a set of local context information in the form of semantic matedata. In addition, Section 4.7 introduces an annotation engine as an extension to traditional Web editor to embody the annotation process. However, Section 4.2.4 argues that there are several alternatives to annotate $CSCs$. Also, the annotation process is still a complex process, as Web authors are usually non-experts.

The goal of this chapter is to address these two issues. Hence, it initially evaluates several annotation alternatives. As we will see, our evaluation is based on several criteria such as the required synchronization effort and the level of inter-operability among users' applications. Next, we introduce an interactive annotation process in order to assist authors to specify their contexts and to annotate each type of $CSCs$ with a suitable context information. Finally, the internal structure of the introduced annotation engine are described.

The rest of this chapter is structured as follows. The annotation alternatives are presented in Section 5.2, and the interactive annotation process is detailed in Section 5.3. Finally, Section 5.4 presents the architecture the of the annotation engine.

5.2 Web Annotation Alternatives

The main two annotation techniques used in the Web domain are called external and internal annotations, as already mentioned in Section 2.3. With external annotation, semantic metadata is represented and stored in an external annotation document. Then, these semantic metadata refer to a part of a document (typically an XHTML tag) that is annotated using a pointing language such as Xpointer[1].

The main motivation behind external annotation is twofold. First, it provides a way to annotate already-published HTML documents without changing them and to annotate new ones without introducing new elements into their document-type definition (DTD). Secondly, one annotation document can be reused for annotating multiple Web documents. This is useful for annotating (already-published) Web documents, or parts of them, that have common semantics and structures such as calender events and products specifications. However, this technique requires additional work to synchronize the annotation document with the annotated document [52, 59, 66].

In contrast, internal annotation localizes (embeds) semantic metadata inside a document element (i.e., XHTML tag) that represents a Web content to be annotated. As we will see in the next sub-sections, internal annotation remains simple, and does not require additional synchronization work, since semantic metadata are directly embedded in the document to be annotated. However, internal annotation requires additional work for annotating already-published Web documents. Indeed, each XHTML tag that represents a Web content to be annotated should be edited separately to embed the annotation [3, 56].

On the basis of these two techniques, this section discusses and evaluates four alternatives to annotate $CSCs$ with semantic metadata (i.e., a concept S and a set of context attributes C). Our evaluation focuses on two aspects. The first aspect considers the way semantic metadata are stored and associated with $CSCs$. The second one considers when and where the values of dynamic context attributes are inferred from the values of static attributes. Also, it discusses the consequences of these two aspects on the Web 2.0 use cases. For consistency purpose, the example presented in Section 1.2.1 is used in order to illustrate our evaluation.

5.2.1 External Annotation

The first alternative is to rely on the idea of external annotation technique directly. In this sense, the semantic metadata related to a concept S and static/specified dynamic

[1] `http://www.w3.org/TR/WD-xptr`

context attributes C are represented and stored in an external document. Then, an annotation pointer from this document is used to refer to a part of a Web document that represents a CSC to be annotated (i.e., an XHTML tag). Technically, the semantic metadata are represented inside an RDF statement as a set of RDF attribute-value pairs in an RDF/XML document[2]. The subject of this statement is an Xpointer that refers to such XHTML tag in the annotated document that represents a Web content to be annotated. Finally, a meta tag that includes the URL of an annotation document is added to the annotated document in order to link it with the annotation document explicitly.

In this setting, a reader's application interprets annotated $CSCs$ at run-time as follows. First, it utilizes the URL(s) included in meta tags to locate the annotation document(s). Secondly, it uses the Xpointer expressions to locate such XHTML tags that represent annotated $CSCs$. Thirdly, the type of each annotated CSC and its corresponding static/specified dynamic attributes are extracted from the RDF attributes-value pairs. Finally, the values of non-specified dynamic attributes are inferred from the values of other static/specified dynamic attributes.

Example

To better understand this alternative and its consequences on the Web 2.0 use cases, Figure 5.1 illustrates how to annotate the date contents involved in our example using this alternative[3]. Here, the date contents in pages A and B are annotated as semantic objects. Then, they are aggregated to page C.

With respect to page B, the annotation of the date is represented as an RDF statement and stored in a separate document (i.e., the A-Cxt2 document). The subject of this statement (i.e., the value of the *rdf:about*) is an Xpointer expression. The first part of this expression, inside *span*, refers to the second $\langle div \rangle$ tag in page B. This tag represents the date content '2009-09-10'. The RDF property *rdf:type='dc:date'* represents the date concept S and the other RDF constructs represent the set of static attributes C that are relate to the Canadian author. On the other side, a meta tag is added into the annotated document (i.e., page B) in order to link it with the annotation document explicitly.

With respect to page A, when the the American author updates the date content at task T2, he (should) update it according to his local context[4]. Hence, the annotation

[2] `http://www.w3.org/TR/REC-rdf-syntax/`

[3] This example based on W3C external annotation note: `http://www.w3.org/TR/annot/`.

[4] Due to the similarity with the above, the date content before update task and its corresponding

should be updated also. To do this, two steps are required. First, deleting an annotation pointer that refers to this date from the annotation document that represents the local context of the original author. Secondly, adding an annotation pointer that refers to this content from the annotation document that represents the local context of the American author (i.e., the A-Cxt1 document).

Afterwards, the aggregation of the annotated date from page A to page C requires the following steps to preserve their consistency. First, adding another Xpointer expression to the subject of the RDF statement stored in the A-Cxt1 document. This pointer is represented inside *span* expression (i.e., the second part) and refers to div[20] that represents the aggregated date CSC. Secondly, adding new meta tag to page C in order to link it with the annotation document explicitly. Finally, updating (re-mapping) all pointers that refer to all annotated *div* elements after *div*[20]. The same steps are needed to aggregate the annotated date from page B to the div[30] inside page C. It is worth noting that the re-mapping step is also required when new element (e.g., div tag) is added or existing one is deleted.

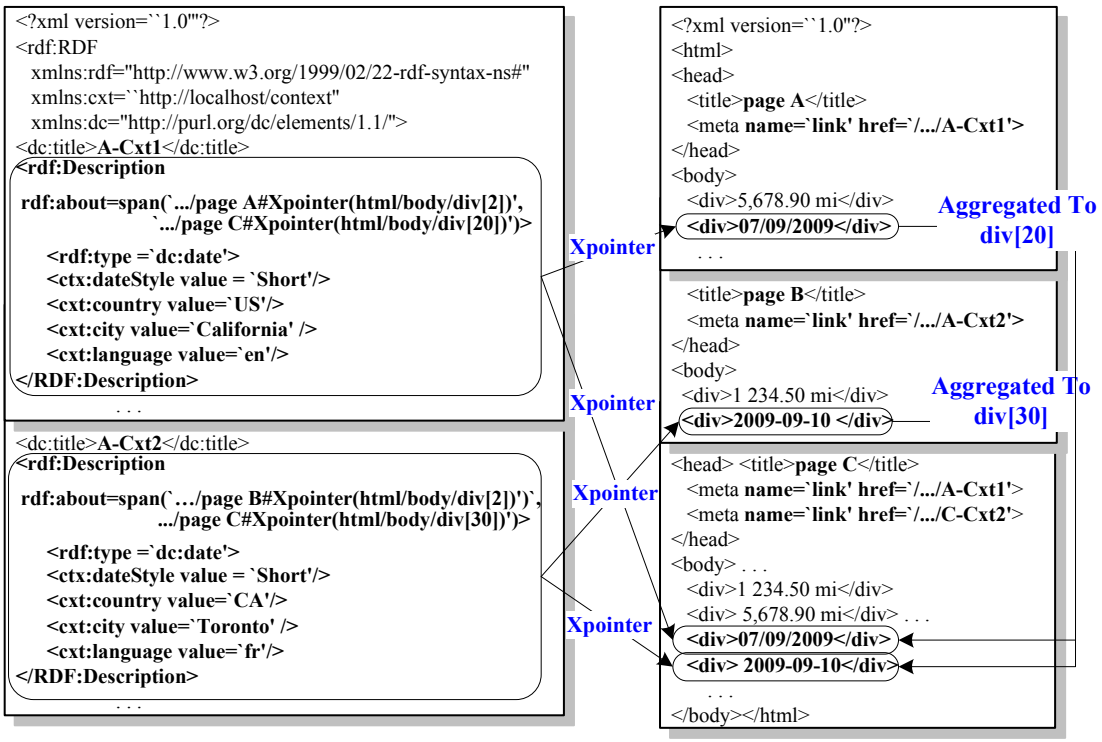

Figure 5.1: External annotation using RDF/XML and Xpointer specifications

annotation are not presented in this example.

Discussion

External annotation faces significant limitations with respect to creation/deletion, update, and aggregation of Web contents and semantic metadata. As already mentioned above, this technique requires additional work to synchronize the annotation document with the annotated document. This is because Xpointers refer to annotated elements based on their paths in the annotated document. Also, external annotation leads to problems when aggregating contents. First, references to the document structure are modified; and second, aggregation may imply context changes, and as the external annotation documents are attached to the original contents, they should not be updated when the context changes due to aggregation in different contexts.

5.2.2 Internal Annotation Using Semantic Metadata URI

The second alternative is to represent and store author's local context in a separate annotation document. This document represents local context attributes and their values, at least static attributes, inside an RDF statement as a set of RDF attribute-value pairs. Additionally, it utilizes the RDFa-based internal annotation techniques to embed the URI of an annotation document and a concept S inside such XHTML tag that represents a CSC to be annotated. In other words, this alternative considers an author's local context as a single entity that is identified by the URI of its corresponding annotation document.

In this setting, a reader's application interprets an annotated CSC at run-time as follows. First, it uses the embedded concept S to identify the type of this CSC and its corresponding context attributes. Then, it uses the embedded URI to extract the values of context attributes that are specified in the corresponding annotation document (static and dynamic attributes). Finally, the other dynamic attributes are inferred from the values of the specified ones.

Example

Figure 5.2 illustrates the use of this alternative to annotate the date contents involved in our example. As it is shown, the local contexts of the American and the Canadian authors are represented and stored inside the A-Cxt1 and the A-Cxt2 documents, respectively.

In addition, the date CSC involved in page B is annotated using this alternative as follows. First, it is identified as a date *SemObj* using the *about ='#D2'* RDFa expression. Next, the *dc:date* is embedded inside the tag as a concept S. Finally, the URI of the A-Cxt2 document is also embedded in order to refer to the Canadian

author's context. The date *CSC* that is created by the British author at task T1 is annotated in similar way[5]

When the the American author updates the date *CSC* involved in page A at task T3, he (should) update it according to his local context. So, the annotation should be updated also. To do so, the URI of the annotation document related to the British author is replaced by the URI of the annotation document related to the American author (i.e., `http://.../A-Cxt1/`). Finally, the annotated dates from pages A and B are aggregated to page C together with their annotation like *"copy and past"*.

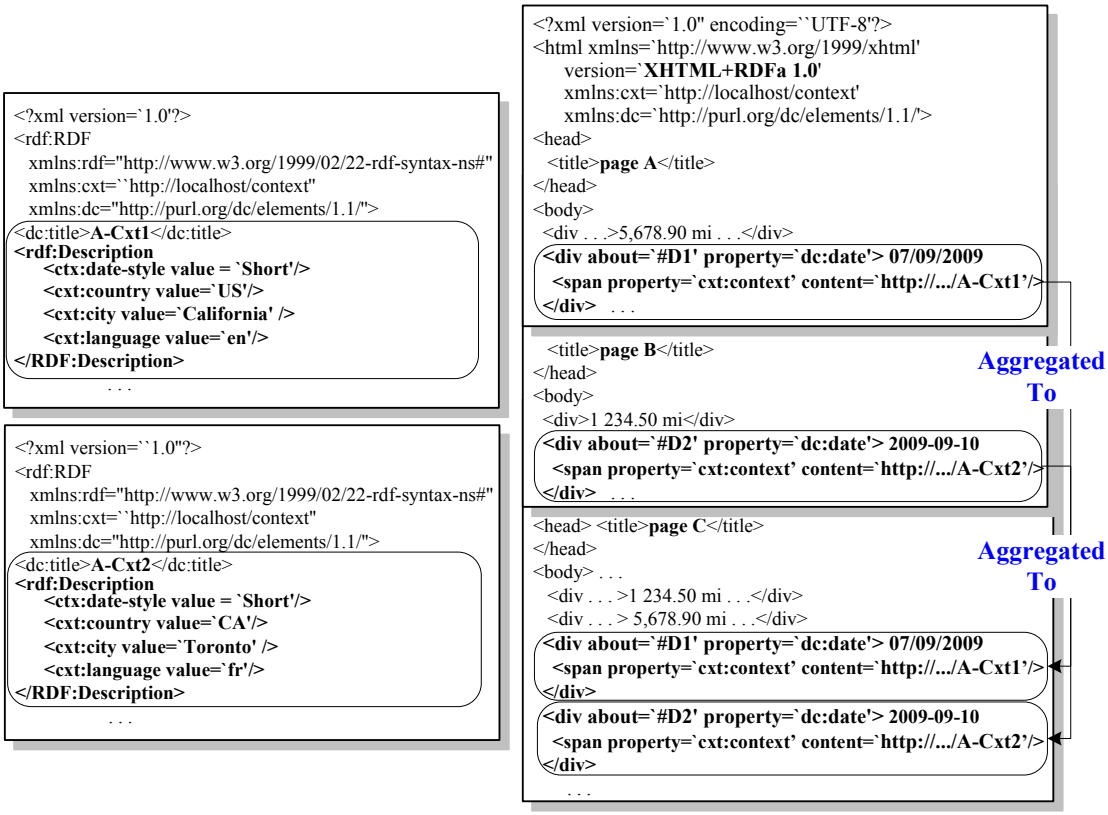

Figure 5.2: Internal annotation using semantic metadata URI

Discussion

Unlike the external annotation, this alternative does not require additional synchronization work. Also, context information stored in separate annotation documents can

[5]Due to the similarity with the above, this date and its corresponding annotation document are not presented in this example.

be reused to annotate many *CSCs* that are involved the same or different XHTML documents. This avoids a redundancy problem. For instance, if two or more *CSCs* are created by the American author, then the URI of the A-Cxt1 document has to be embedded in the corresponding XHTML elements only.

However, this alternative face a significant limitation related to the evolution of authors' local contexts. Indeed, the local contexts of users could be changed over time. For instance, the American author above may change his living city from *California to London*. Hence, he logically has to use the value of London time zone to annotate new date *CSCs* or update existing ones. In contrast, the old value of time zone (i.e., California time zone) must be preserved for the date *CSCs* that already created and annotated before. Since the local context of an author is embedded in an annotated document as a single entity via the URI of an annotation document, two different versions of this author context are needed to consider this scenario. Furthermore, each future change on an author's context needs a new version. Hence, this alternative require additional work to manage many versions of an author context. Note that, the aforementioned external annotation face the same evolution problem.

5.2.3 Inline Internal Annotation Using Static Context Attributes

This alternative keeps representation and storage of an author's local context in a separate annotation document. However, it utilizes the RDFa-based internal annotation to embed a concept S and a set of context attributes in such XHTML tag that represents a CSC to be annotated. Here, the embedded context attributes include static attributes and their values corresponding to the annotated CSC. Also, if an author needs specific values for one or more dynamic attributes, then the specified dynamic attributes are embedded instead.

To interpret an annotated CSC in this setting, a reader's application uses the embedded concept S to identify the type of this CSC. Then, it utilizes the relations between contexts attributes that are described in the LCO in order to infer the values of non-specified dynamic attributes from the other context attributes embedded inside the XHTML tag (see Section 4.5 and Section 4.6).

Example

As it is shown in Figure 5.3, this example illustrates the annotation of the date contents in the above example using this alternative. Here, it is assumed that the local contexts of the American and Canadian authors are represented in the A-Cxt1 and A-Cxt2

documents, respectively[6]. Then, these documents are utilized in this example as follows.

In page B, the date *CSC* is identified as a date *SemObj* using the *about ='#D2'* RDFa expression. Next, The RDFa attribute *property='dc:date'* represents the date concept S and the date content '2009-09-10' represents the value V of this date. In the inner $\langle span \rangle$ tags, the set of RDFa *property* attributes represent the static context attributes related to date semantic object, and the RDFa *content* attributes represent the values of the these attributes related to the Canadian author. Here, the values of static attributes are copied from the A-Cxt2 document at annotation time.

Like the above example, when the the American author updates the date *CSC* involved in page A at task T3, its annotation should be updated also. In this setting, the values of context attributes related to the British author are replaced by the values of context attributes related the the American author. Finally, the annotated dates from pages A and B are aggregated to page C together with their annotation like *"copy and past"*.

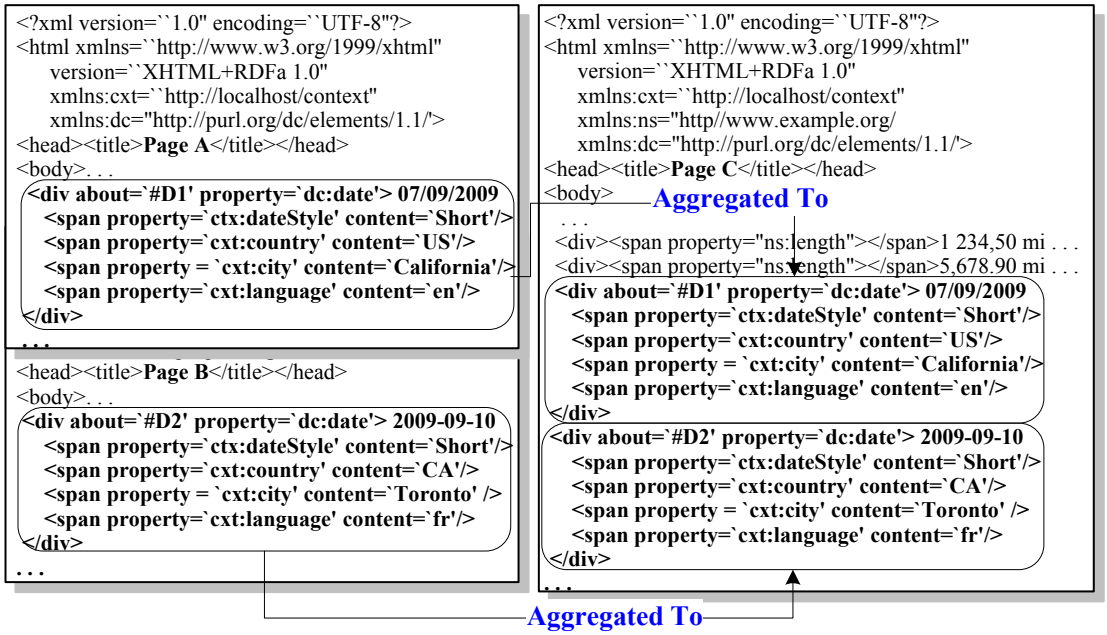

Figure 5.3: Inline internal annotation using static context attributes

[6]These documents are shown in Figure 5.2 above.

Discussion

This alternative avoids the problem of local context evolution, since the values of context attributes are embedded in the annotated document at annotation time. Practically, this means that the already annotated $CSCs$ does not affected by any change that is carried ont on A-Cxt1 document. In contrast, this change will be reflected on the annotation of new $CSCs$ or on the annotation updated $CSCs$. In other words, a single version of local context is stored for each user at a time. This context is utilized to annotate $CSCs$ that are created or updated at this time.

However, this alternative and the other aforementioned alternatives rely on readers' applications to infer the values of dynamic context attributes. This implies several issues. First, the level 3 of common conceptualization that is identified in Section 4.3.3 have to be reached among users' applications. As already discussed in that section, this level restricts the extensibility of the LCO. Secondly, the inference of the dynamic attributes values add more overhead on readers' applications. Note that, there is an overhead already added on these applications, as they have to adapt the values of $SemObjs$ according to their readers' contexts.

In addition, this also does not comply well with the semantic visibility principle discussed in Section 2.3.3. Based on this principle, context information can be made visible to readers in case the annotated $CSCs$ can not be adapted to their local contexts (e.g., readers' applications are not extended with adaptation engines). In this setting, there is no great benefit to make static context attributes visible to readers. In contrast, it is better to use these attributes to infer other context attributes, and then make the latter visible to readers. For instance, it is better to make the time zones values of the date $CSCs$ visible in the above example rather than the countries and cities values.

5.2.4 Inline Annotation Using Minimum Context Attributes

The last alternative is relatively similar to the third one. However, instead of embedding static context attributes, it embeds the minimum set of context attributes and their values in such XHTML tag that represents a CSC to be annotated. This implies that an author's application is responsible of inferring the values of dynamic context attributes from the values of context attributes stored in the annotation document related to this author (e.g., A-Cxt1 document). On the other side, a reader's application is responsible to infer the values of dynamic context attributes related to this reader. Figure 5.4 presents an example to illustrate the annotation of the date contents in the above example using this alternative. This example is similar to the example presented in Figure 5.3.

However, it embeds the minimum set of context attributes corresponding to these date $CSCs$ instead of static attributes.

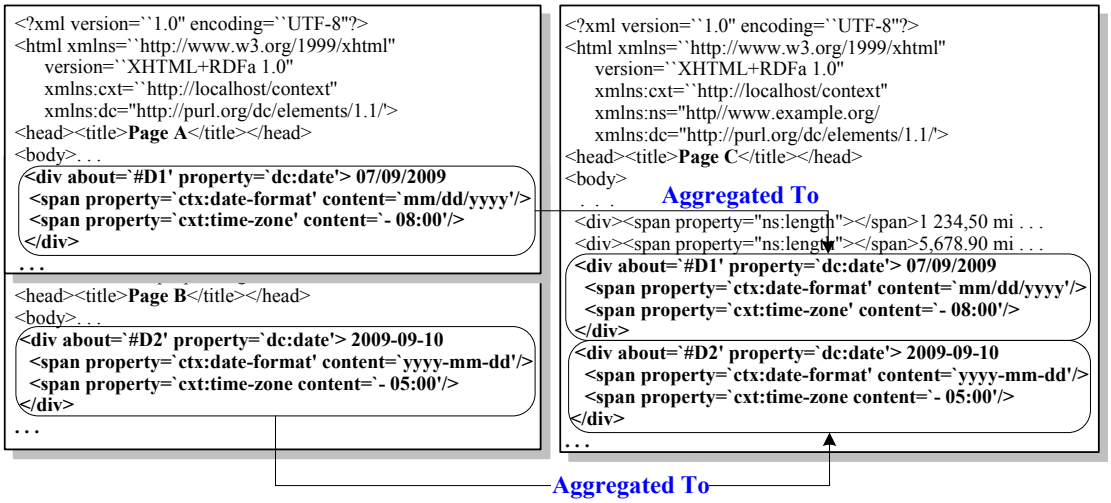

Figure 5.4: Inline internal annotation using a minimum context attributes

Discussion

The main difference of this alternative among other alternatives lies into when and where dynamic attributes are inferred. The first three alternatives recommend to infer the dynamic attributes related to both authors and readers at browsing time and by readers' applications. However, this alternative recommends to infer the dynamic attributes related to authors at annotation time and by authors' applications. Also, the dynamic attributes related to readers is recommenced to be inferred at browsing time and by readers' applications. In addition, this alternative utilizes the RDFa specification to identify the relations between S, CSC, and the minimum set of context attributes. This implies that users' applications have to reach to the level 2 of common conceptualization that is identified in Section 4.3.3 (i.e., they have to agree on a concept S and the minimum set of context attributes related to each type of sematic object). Moreover, if authors' and readers' applications do not reach to the level two of common conceptualization, still there an opportunity to make the context information such as date formats and time zones visible to readers. This helps readers to correctly interpret these $CSCs$ even if it is not possible to adapt them according to their local contexts.

5.2.5 Web Annotation Alternatives: Conclusion

It is obvious that the fourth annotation alternative is the best alternative to annotate *CSCs* as semantic objects. It does not require additional synchronization effort to create, delete, or aggregate *CSCs* in Web 2.0 use cases. Also, the evolution of authors' contexts overtime is easy to manage. Moreover, users' applications require to reach to the level 2 of common conceptualization in order to ensure a consistent inter-operability of semantic objects. Finally, it complies well with the semantic visibility principle. Table 5.1 summaries our evaluation of the annotation alternatives.

Annotation Alternatives	**External Annotation**	**Internal Using URI**	**Internal Using static Att.**	**Internal Using Minimum Att.**
Synchronization Effort	Yes	No	No	No
Users' Context Evolution	Hard to manage	Hard to manage	Easy to manage	Easy to manage
Inter-operability level required	Third level	Third level	Third level	Second level
Semantic visibility of context	No	No	Partial	Yes

Table 5.1: Evaluation summary of the annotation alternatives.

5.3 Interactive Web Annotation Process

Having adopted the forth annotation alternative, the complexity that Web authors could face when they annotate *CSCs* has to be addressed. As already mentioned many times, Web authors need additional effort to annotate *CSCs* correctly or even they could encounter errors during this process. This is because they are mostly non-experts and do not know the relations between *CSCs* and local context information. Also, they do not have theoretical and technical knowledge about the annotation process.

This section introduces an interactive annotation process that aims at assisting Web authors to annotate *CSCs*. Basically, Web authors can utilize this process to specify and store their contexts information, and also to annotate each *CSC* with a corresponding concept S and a corresponding minimum set of context attributes.

As shown in Figure 5.5, our annotation process consists of one pre-annotation task (i.e., Task 1) and four annotation tasks. The following subsections describe each of these tasks in more details.

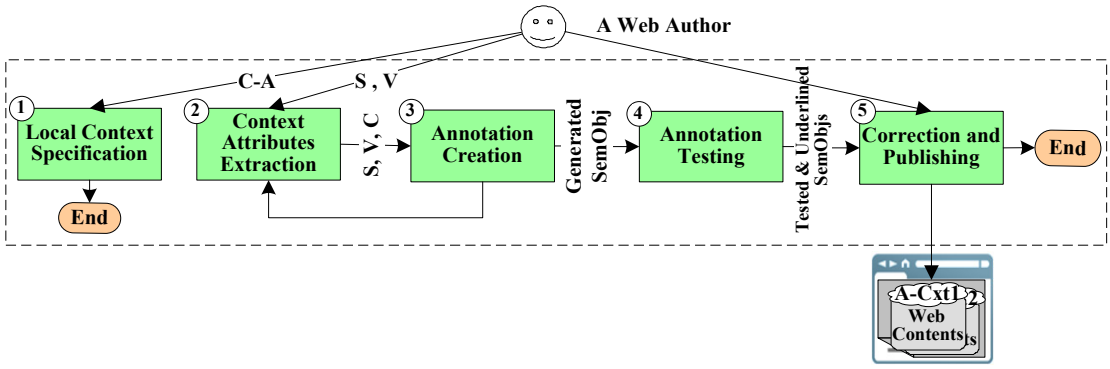

Figure 5.5: Overview of the annotation process

5.3.1 Local Context Specification

Prior annotating $CSCs$, a Web author has to specify his local context information. Also, this information has to be stored in the A-Cxt document related to him. This task assists an author to do this as follows:

- An author has to specify static context attributes at minimum. Also, he can specify one or more dynamic attributes if there is a specific need for unusual/specific dynamic values.
- Then, the specified values are stored in the A-Cxt document as instances of context attributes specified in the LCO (see Section 4.5).

Figure 5.7[7] presents a flowchart to illustrate this task. Recall that, the categorization of local context information into static and dynamic attributes assist authors to specify their local context as the values of static attributes can be easily specified (see Section 4.3.1).

5.3.2 Context Attributes Extraction

A CSC has to be annotated with a concept S and a minimum set of context attributes C, so that it is represented as a semantic object. This is interactively performed in this task and in the following task (i.e., annotation creation). Indeed, this task assists an author to identify the relation between a CSC and the real world aspect it describes (i.e., a concepts S). Also, it hides the relations between a CSC and its corresponding context attributes as follows (see Figure 5.7):

[7]The symbol * indicates that the specification of dynamic attributes is optional.

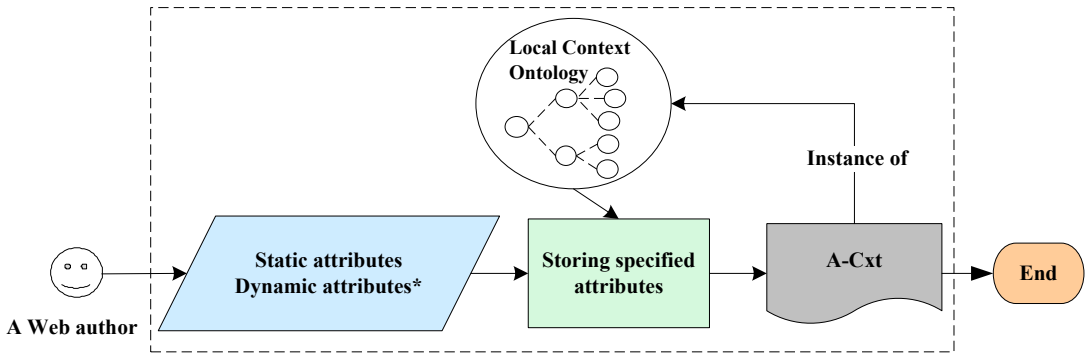

Figure 5.6: Annotation process: local context specification flowchart

- An author needs to identify (i.e., highlight) the value V of this CSC during content creation or update. Then, he needs to identify a concept S that it adhere to.
- Upon the identification of S, this task extracts the set of context attributes from the context ontology based on their relations with the selected S. These relations are described in more details in Section 4.6.
- Afterwards, the value of the extracted context attributes are specified. Here, the values of static context attributes and the specified dynamic context attributes (if any) are extracted from the A-Cxt document related to this author. Also, the values of non-specified dynamic attributes are inferred from the values of the specified attributes. Recall that, it is considered that this inference is based on the relations between static and dynamic attributes described in the LCO.
- Finally, the identified S, V, and the minimum set of context attributes C together with their values are utilized as inputs to the annotation creation task below.

5.3.3 Annotation Creation

Using the inputs received from the context attributes extraction task, this process annotates a CSC as a $SemObj$ as follows:

- A $SemObj$ instance is built in the memory based on the received concept S. Then, the received concept S, the value V, and the minimum set of context attributes C are assigned to this $SemObj$.
- Next, an XHTML+RDFa representation of this $SemObj$ is generated using the RDFa specification.

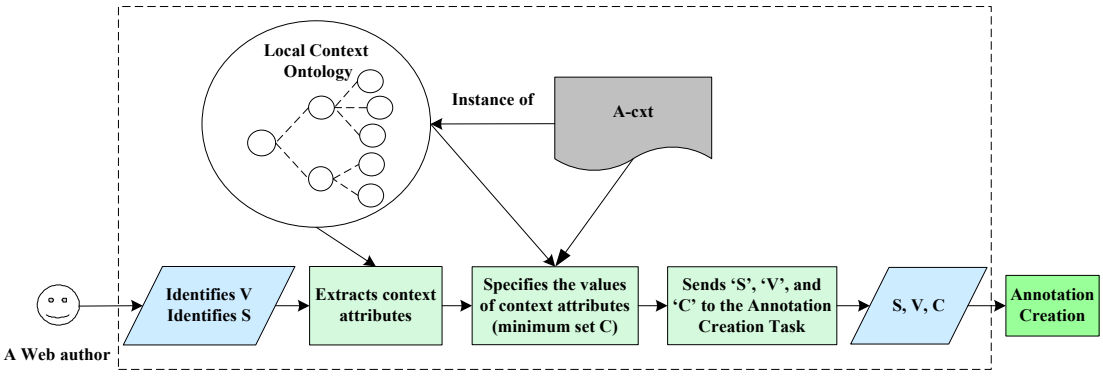

Figure 5.7: Annotation process: context attributes extraction flowchart

- Afterwards, the value V of a CSC is replaced by the generated representation.

- Finally, the generated representation (i.e., $SemObj$) is utilized as input to the annotation testing task below.

Figure 5.8 presents a flowchart to illustrate this task. This task and the previous one can be repeated by an author in order to annotate other $CSCs$. Note that, an author might annotate a CSC created and annotated by another author. In this setting, the value V of this CSC together with its previous annotation is replaced by the new generated representation. This ensures that the annotation of $CSCs$ annotated by other authors are updated according to the author's context who update them.

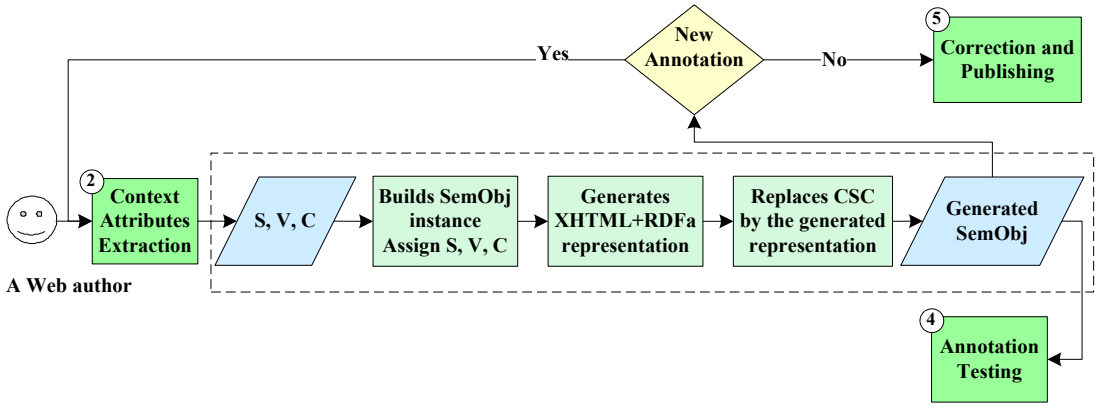

Figure 5.8: Annotation process: annotation creation flowchart

5.3.4 Annotation Testing

The role of this task is to detect the potential errors that authors could perform when they annotate $CSCs$. To this end, it scans each generated $SemObjs$ in the background as follows (see Figure 5.9):

- A $SemObj$ instance called $testerSemObj$ is built for each generated $SemObj$ received from the annotation creation task. The $testerSemObj$ takes the concept S and the context C of the generated $SemObj$ as parameters and generates a test value TV.

- Then, the generated $SemObj$ is compared with the tester $SemObj$ as follows:

  1. The value V is compared with the value TV and provides a warning message as long as the former does not comply with TV (like a smart tag in Microsoft Word). For example, if an author, by mistake, annotates a length CSC with a date concept, then the generated $SemObj$ will be underlined and provides a warning message (e.g., this content is not a date).
  2. The value V is compared with the context attributes extracted/inferred from the A-Cxt document, and provides a warning message if V does not comply with one of them. For example, a warning message is provided if an author represents a date CSC in a way that violates the value of the date-format attribute extracted/inferred from his related A-Cxt document.

- The output is a tested $SemObj$, either correct or incorrect (i.e., underlined) object.

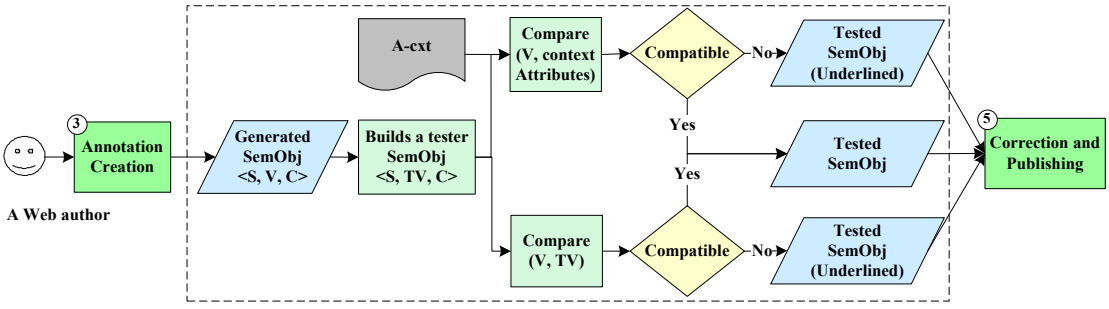

Figure 5.9: Annotation process: annotation testing flowchart

5.3.5 Correction and Publishing

The role of this task is to assist authors for correcting the annotation errors that are detected in the previous tasks, and for publishing the annotated $CSCs$ to Web 2.0 pages. This is interactively performed as follows (see Figure 5.10):

- An author has to correct the underlined $SemObjs$. For example, he needs to correct the annotation of the length CSC, in the above example, with the length concept (instead of the date concept).
- Then, he publishes the annotated $CSCs$, and also the other Web contents he creates or updates, to the intended Web 2.0 page.

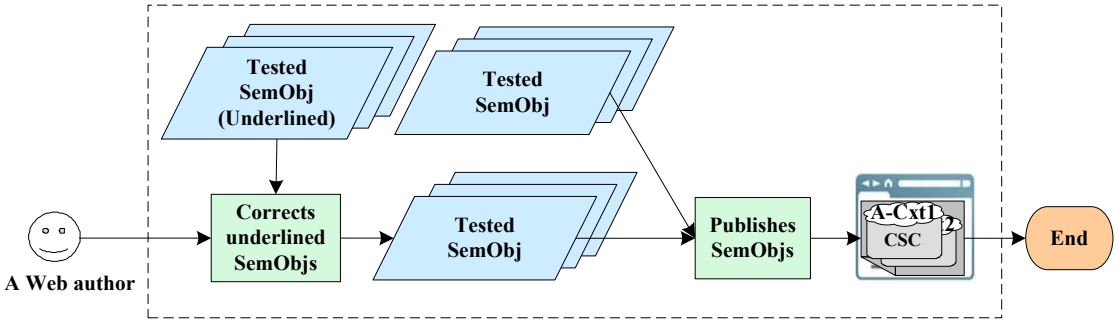

Figure 5.10: Annotation process: correction and publishing flowchart

5.3.6 Conclusion and Suggested Extensions

The above tasks provide the means to annotate $CSCs$ in an interactive and easy manner. First, context attributes that are specified in the A-Cxt document can be reused to annotate Web contents at different authoring times. Second, Task 2 enables authors annotating $CSCs$ as easy as formatting text in word processors [77]. Furthermore, authors do not need to know the relations between the $CSCs$ and the local context attributes, since the context attributes and their values are extracted automatically based on the concept S. At the same time our approach is flexible as advanced authors still have the possibility to override dynamic inferred attributes in the annotation. Third, creation of semantic annotation, in Task 3, hides the technical complexities of the RDFa syntax. In addition, it reduces the annotation efforts and the number of annotation errors that could be performed by Web authors. Finally, testing semantic objects, in Task 4, assists authors into correctly authoring $CSCs$ and therefore reduces the potential errors that they could perform.

The annotation process can be extended in one of the following directions. First, it can be extended with information extraction, together with semantic-aware auto-complete interface [61]. Information extraction is used to match typed $CSCs$, at authoring time, with one of the concepts S based on predefined concept patterns for example. If the matching task succeeds, then the semantic auto-complete interface will recommend this concept to the author. This increases the willingness of authors for annotating $CSCs$. Also, it helps authors knowing which $CSCs$ need to be annotated. Second, providers (i.e., Web sites designers or administrators) can relate concepts S and context attributes C to *annotation templates*, such as event and product templates. Based on an author's context, these templates are generated upon an author's request, and the typed (filled) $CSCs$ are annotated accordingly. This is useful for annotating structured contents [69]. Third, the annotation testing task can reuse the information extraction technique to check if there is a CSC that matches one of the concept S. If so, then it provides a warning message to the author in order to confirm or deny this matching. This enhances the annotation results, since authors could forget annotating some $CSCs$.

5.4 Annotation Engine: Architecture and Prototype

With respect to the architecture perspective, Section 4.7 introduces an annotation engine as an extension to traditional Web editors. The role of this engine is to embody the aforementioned annotation process. This section describes the internal structure of this engine. Also, it presents a prototype which proves the validity of the proposed annotation process.

5.4.1 Internal Structure

Figure 5.11 depicts two views of the annotation engine: process and architecture views. The *process view* is already detailed in the previous section. The architecture view illustrates the internal structure of this engine. This view encompasses two modules: interface and annotation functions modules. Also, it utilizes the concept description layer that is described in the general architecture (See Section 4.7).

The role of the *interface* module is to enable Web authors to specify their local contexts and annotate $CSCs$ with suitable semantic metadata as follows. Therefore, the interface of a traditional Web editor (i.e., the Web editing interface component) is extended with the Local Context form (or L-C form for short) and the concepts menu components. The *L-C form* allows an author to specify his local context information.

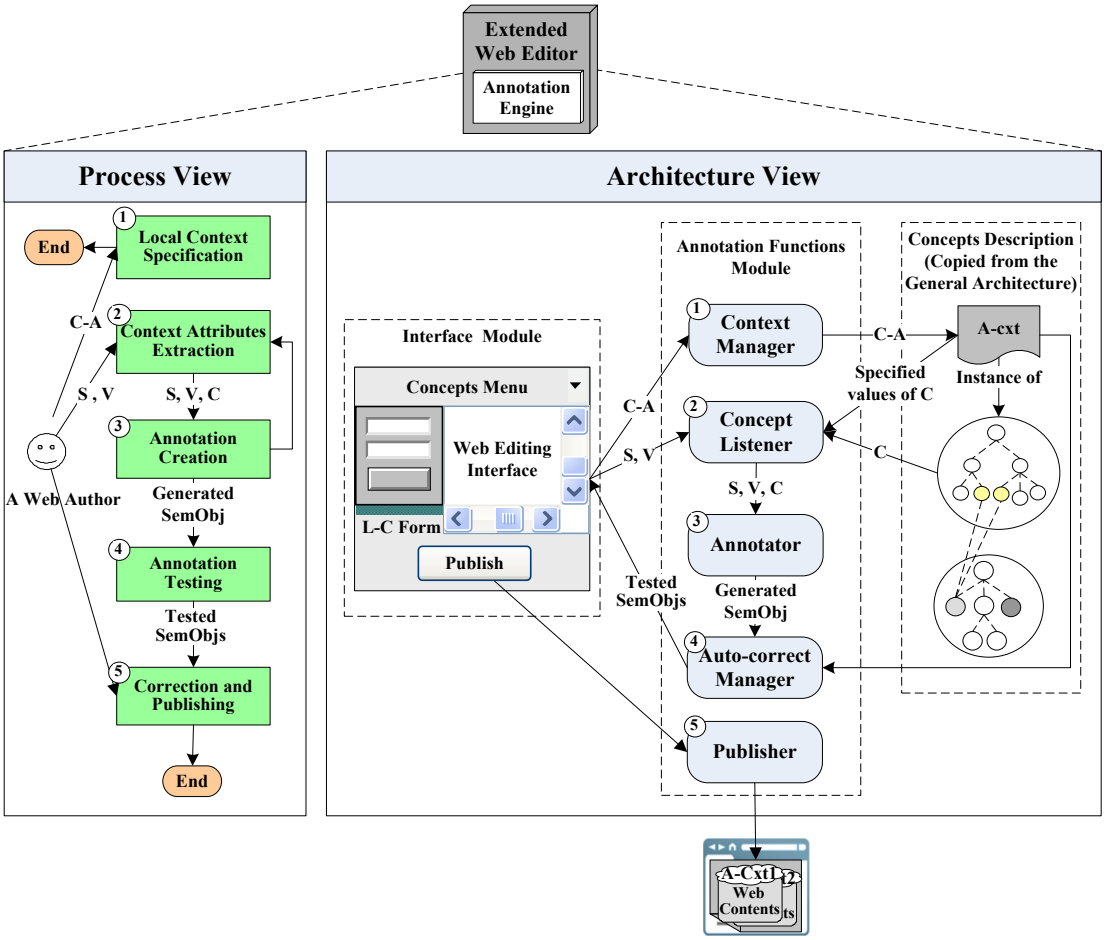

Figure 5.11: Annotation engine: internal architecture view

Afterwards, the *concepts menu* allows this author to select a suitable concept S for annotating a CSC who identifies it in the Web editing interface.

The role of the *annotation functions* module is to embody the aforementioned annotation tasks. Indeed, each annotation task, shown in the process view, is designed as a function in this module[8]. These functions are invoked upon authors' interactions with the interface components. For instance, the context manger function embodies the local context specification task. This function is invoked when an author specifies and submits his context information and stores the specified information in the A-Cxt document related to this author.

[8]In Figure 5.11, the same number is given for each annotation task and its related annotation function.

5.4.2 Web Annotation Prototype

In addition to the annotation engine architecture, we also introduce an annotation engine prototype as a proof-of-concept. Our prototype extends the TinyMCE™WYSIWYG Web editor[9] with our annotation engine as follows. The concepts menu is implemented using TinyMCE APIs[10]. The L-C form is implemented using an HTML form. This form is also supplemented with an illustrative example to assist authors to specify their local context information correctly.

With respect to annotation functions module, we utilize java™and javascript™APIs to implement them. Indeed, java APIs are used to store context information in an author's A-Cxt document. Also, when an author selects a concept S from the concepts menu, java APIs are also used to extract the corresponding context attributes and their values from the LCO and A-Cxt document. Javascript APIs are utilized to annotate the identified CSC with the selected concept S and the extracted context attributes C. Also , they are used to test the correctness of the annotated $CSCs$ and publish the contents involved in the editing interface to a Web 2.0 site. In other words, java APIs are utilized to implement the context manager and the concept listener functions. Also, Javascript APIs are utilized to implement the annotator, auto-correct manager, and the publisher functions. Figure 5.12 presents a screenshot of our prototype and illustrates how the date content from our scenario during Task T3 can be annotated.

[9] `http://tinymce.moxiecode.com/`

[10] TinyMCE APIs are a set of javascript APIs to dedicated to extends the TinyMCE editor.

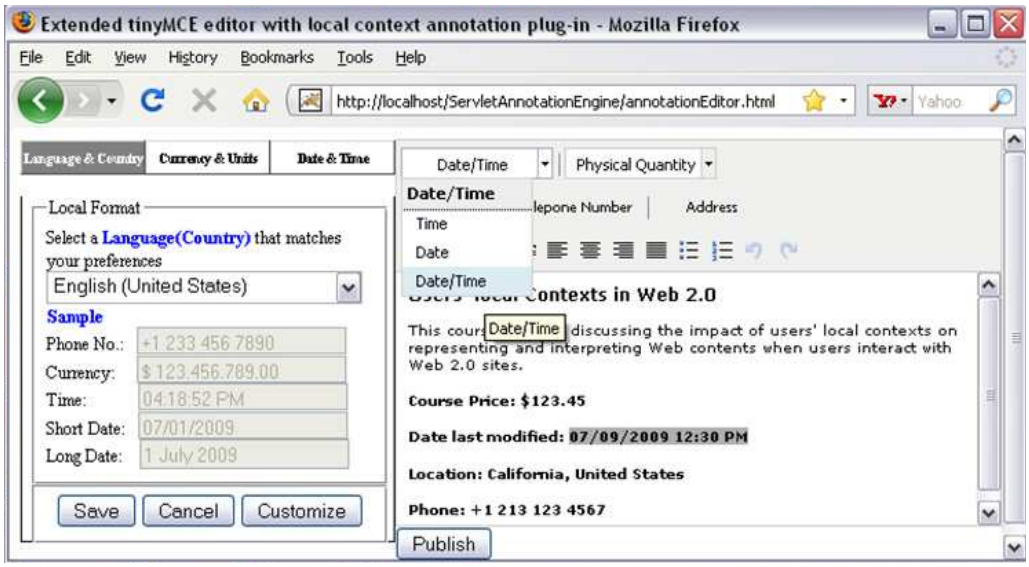

Figure 5.12: A screenshot of the extended Web editor.

Chapter 6

Web Adaptation

6.1 Introduction

The term Web adaptation is identified in Section 1.3.3 as a process of adapting $CSCs$ according to readers local contexts. The goal of this adaptation is to address the *incorrectness and inefficiency* problems that Web readers could encounter when they interpret $CSCs$ that are represented according to their authors' local contexts (see Section 1.2.2). In addition, Section 4.2 concludes that the annotation of $CSCs$ with their authors' contexts at creation/update time and the adaptation of the annotated $CSCs$ according to their readers' local contexts at browsing time is the best design alternative with respect to the Web 2.0 use cases. Accordingly, an adaptation engine is introduced in Section 4.7 as an extension to readers' applications to embody the adaptation process.

On the basis of the adopted design alternative, this chapter aims at discussing the design of Web adaptation. Hence, Section 6.2 initially describes the notion of adaptation functions and the requirements to apply this notion in our approach. Next, a set of adaptation functions for each type of $SemObjs$ are identified in Section 6.3. Based on the latter, the process for adapting $SemObjs$ involved in a Web page is detailed in Section 6.4. Finally, the internal structure of the introduced adaptation engine is described in Section 6.5.

6.2 Web Adaptation: Theory and Requirements

Our adaptation process mainly relies on the adaptation functions notion. Basically, the latter adapts the value of a $SemObj$ from one context to another based on the semantic metadata that enrich this object. Hence, it complements the semantic object notion.

This section presents the adaptation functions notion from a theoretical viewpoint. Afterwards, it discusses the requirements to practically apply it in our approach, taking

in consideration the design decisions that were already adopted in the previous chapters.

6.2.1 Adaptation Functions

Using the semantic object notion to enrich $CSCs$ with semantic metadata (i.e., a concept S and context attributes C) provides a mean for readers' applications to adapt them from their authors' contexts to their readers' contexts. Indeed, the concept S allows a reader's application to identify the type of such $SemObj$ to be adapted, since this concept specifies the relation between a CSC and the real world aspect it describes. In addition, the set of context attributes that enrich this $SemObj$ describe the context information used by its author to represent this CSC (see Section 4.3). In accordance, one or more adaptation functions can be identified to adapt the value of this $SemObj$ according to a reader's context.

Theoretically, for a given $SemObj = <$ S, V, C $>$, such that C $= \{c_1, c_2, ..., c_n\}$ (n $\in \mathbb{N}$) is a set of context attributes corresponding to author's local context, a set of adaptation functions $F = \{f_1, f_2, \ldots, f_m\}$ ((m $\in \mathbb{N}$) can be identified. These functions adapt the $SemObj$ according to a given set of context attributes $C' = \{c'_1, c'_2, ..., c'_n\}$ corresponding to a reader's local context. This adaptation is performed as follows:

- The concept S is used to identify the set of adaptation functions F corresponding to this $SemObj$.
- Each adaptation function f_i involved in F takes the value V and a subset of context attributes from both C and C' sets as parameters, and adapts V according to the reader's context.
- The result of applying the set of adaptation function F is a new semantic object $SemObj' =$ < S, V', C' >.

$SemObj'$ is semantically equivalent to the original semantic object (i.e., $SemObj$), but its value is represented according to the reader's context C'. In other words, both of them describe the same CSC, but they are represented according to different local contexts. The output of applying the adaptation functions F can be formalized as follows:

$$\boldsymbol{F}\ (<S,\ V,\ C>, C') = <S, V', C'>$$

It is worth noting that our approach adopts the idea of adaptation functions from the approaches that have utilized the semantic object notion [24, 82, 100]. These approaches discuss several aspects related to adaptation functions in more details.

Example

Concerning the date CSC updated by the American author in our example, the following tuple presents it as date $SemObj$[1]. Here, the *Date* refers to concept S. Also, the *date-format* and the *time-zone* refer to the corresponding minimum set of context attributes used by the American author:

Date SemObj =<*Date, 07/09/2009, date-format =‘mm/dd/yyyy’, time-zone=‘-08:00’*>

To adapt this date $SemObj$ according to the French reader's context, two adaptation functions are needed. The first function is needed to handle the time zone difference (called time-zone convertor). This function takes the date value (i.e., 07/09/2009) and the time zone values related to the American and the French users as parameters. Afterwards, it adapts the date value from the California time zone to the Paris time zone[2]. The second function is needed to handle the date format difference (called date/time formatter). Like the first function, it takes the date values and the date format values related the American and the French users as parameters. Then, it adapts the date value from the American date format to the French date format. The following tuple presents the resulting semantic object after applying these two adaptation functions (i.e., date $SemObj'$):

Date SemObj' =<*Date, 09/07/2009, date-format =‘dd/mm/yyyy’, time-zone=‘+01:00’*>

6.2.2 Web Adaptation Requirements

Practically, several requirements have to be considered in order to utilize the adaptation functions notion in our approaches. Many of these requirements are taken into account when several design decisions are made in the previous chapters. The goal of this section is to identify these requirements, and to refer to the ones that are already considered.

The first intuitive requirement is related to Web readers' contexts. Web readers must be able to specify their local context information easily. This requirement is taken into account in the design of the LCO and the categorization of context attributes into static and dynamic attributes (see Chapter 4).

Secondly, Web adaptation must be able to parse the requested Web page's contents and identify Web contents that are annotated as semantic objects. It also must be

[1] Section 4.3 discusses the representation of this date a $SemObj$ in more details.

[2] As there is 9 hours difference between the author's and reader's local times, the date value is converted to 08/07/2009 as long as the French reader browses this date before 09:00 AM.

able to identify the types of these semantic objects and the context information that are necessary to adapt each type of semantic object. This requirement is also mainly considered in the design strategy introduced in Section 4.3.3. As we will see next, Web adaptation relies on RDFa specification to parse and identify the annotated $CSCs$. In addition, it relies on the common ontologies and the LCO to identify the type of each semantic object and its related minimum set of context attributes.

Thirdly, for each type of semantic object, a set of adaptation functions F have to be defined. In addition, Web adaptation must be able to determine the appropriate adaptation functions that are necessary to adapt each of these semantic objects. The forth and final requirement is to present the adaptation result to Web readers. In the following sections, we will discuss these requirements in more details.

6.3 Semantic Objects and Adaptation Functions

Like the adaptation functions corresponding to the date/time $SemObj$, one or more adaptation functions are required in order to handle the local context aspects related to each type of $SemObjs$ described in Section 4.6. The role of this Section is to define these functions. In other words, this section addresses the first part of the third requirement described above. For consistency reason, each of the following section initially recalls the $SemObj$ type to be adapted. Afterwards, it describes the adaptation functions corresponding to it.

6.3.1 Adaptation of Physical Quantity $SemObjs$

Web users represent each kind of physical quantity $CSCs$ using different (prefixed) measure units and different formats. Accordingly, they are annotated with quantity-dimension, measure-unit, measure-prefix, and quantity-format attributes as $SemObjs$.

The adaptation of a physical quantity $SemObj$ according to such reader's context requires two adaptation functions: measure unit convertor and physical quantity formatter. Each function utilizes a subset of the above attributes and a subset of context attributes related to this reader in order to adapt the value of this $SemObj$ as follows:

- *Measure unit convertor* takes the value V of a quantity $SemObj$ and the values of measure-unit and measure-prefix attributes related to this $SemObj$ and the corresponding values related to the reader as parameters. Next, the value V is adapted according the values of measure-unit and measure-prefix related to the reader. This adaptation can be performed using a mathematical formula or a fixed

exchange rate dedicated to this purpose. For example, the temperature quantities can be adapted from *Celsius to Fahrenheit (or C to F)* using the the following formula: C = (F - 32) * 5/9. Also, the exchange rate between Mile and Meter units is "1 Mile = 1609 Meter".

- *Physical unit formatter* takes the value V and values of *quantity-format* attribute related to this $SemObj$ and its reader. Next, it adapts V according to the reader's quantity format.

6.3.2 Adaptation of Price $SemObjs$

Web users represent price $CSCs$ using different currencies, different sales tax systems/rate, and different formats. Hence, price $CSCs$ are annotated with currency, sales-included, sales-tax-type, sales-tax rate, and price format attributes as semantic objects.

The adaptation of a price $SemObj$ according to a reader's context requires three adaptation functions: currency convertor, sales tax calculator, and price formatter. Each of these functions adapts the value of $SemObj$ as follows:

- *Currency convertor* takes the value V of a price $SemObj$, the value of its representation currency, and the value of currency related to its reader as parameters. Afterwards, it uses the values of both currencies to calculate the currency exchange rates between these two currency. As already mentioned, currency exchange rates are dynamic and they frequently change over time. Hence, we recommend to rely on an online Web service to calculate this rate. Finally, it adapts the value V to its reader's currency by multiplying it by the received exchange rate.

- *Price formatter* adapts the value V according to the reader's price format using the values of *price-format* attributes related to this $SemObj$ and its reader.

- *Sales tax calculator.* As already discussed in Section 3.3.5, dealing with sales taxes on the Web to exchange products and services across countries is still an open issue. Theoretically, sales taxes are imposed on consumers, collected by sellers, and paid to consumers' countries based on their sales tax systems. However, this scenario is not easy to apply in practice and most E-commerce Web sites apply the same sales tax system/rate for all users. And then, we argue that users are interested to know the type of sales tax system that is applied and the sales tax rate as long as the sales tax is included.

In this context, we rely on the scenario used in practice to design our sales tax calculator as follows. This function takes the values of *sales-tax-type, sales-tax-rate, and sales-tax-included* attributes that are involved in the minimum set of context attributes related to this price $SemObj$. Also, it takes the value of *sales-tax-included* attribute related to its reader. Afterwards, it adapts the price value as follows:

- If the value of *sales-tax-included* attribute corresponding to the price $SemObj$ is $False$, then this function notifies the reader that the sales tax is excluded.
- If the values of *sales-tax-included* attribute corresponding to the price $SemObj$ and its reader are $True$, then this function calculates the sales tax value[3]. Afterwards, it presents the price value (i.e., total value including tax value), the calculated tax value, and the type of sales tax (i.e., the value of the *sales-tax-type* attribute).
- If the values of *sales-tax-included* attribute corresponding to the price $SemObj$ is and its reader are $True$ and $False$, respectively, then this function calculates the sales tax value and deducts it from the price value. Then, it presents the price value (i.e., excluding tax value), the calculated tax value, and the type of sales tax.

Note that, the calculated sales tax value is also formatted according to this reader's context.

6.3.3 Adaptation of Telephone Number $SemObjs$

Web users represent telephone number $CSCs$ using different formats. Also, they have to complement these numbers with different prefixes to dial telephone lines inside a country (i.e., over local telephone networks) or across countries (i.e., over global telephone networks). Hence, telephone number $CSCs$ are annotated with phone-format, country-calling-code, national-prefix, and international-prefix attributes as $SemObjs$.

The adaptation of a telephone number $SemObj$ according to a reader's context requires two adaptation functions: prefix(es) adaptor and telephone number formatter. Each of them adapts the telephone number value as follows:

- *Prefixes adaptor* takes the values of country-calling-code attribute corresponding to this $SemObj$ and its reader as parameters. Subsequently, it complements the value V of this $SemObj$ with suitable prefix(es) as follows.

[3]Sales tax value is calculated via multiply the price value V by the sales-tax-rate value.

– If the values of the country-calling-code attribute are the same, then it complements V with the value of the national prefix attribute involved in the annotation.

– If they have different values, then it complements V with the value of international-prefix attribute corresponding to the reader's context and the value of country-calling-code attribute corresponding to this *SemObj*, respectively.

Note that, the prefixes values are added before the telephone numbers values.

- Like the above formatter, *telephone number formatter* takes the values of *phone-format* attributes related to this *SemObj* and its reader, and adapts the value V according to the reader's phone format value.

6.4 Web Adaptation Process

This section discusses our vision on how to achieve the Web adaptation requirements identified above. Basically, we introduce a Web adaptation process that assists readers to specify their contexts information. After that, it automatically adapts the annotated *CSC* according to their contexts at browsing time. In other words, it illustrates the role of the adaption engine introduced in Section 4.7 engine during a Web reader/extended Web browser interaction.

As it is shown in Figure 6.1, our adaptation process consists one pre-adaptation task and five adaptation tasks. These tasks are described in more details in the following subsections.

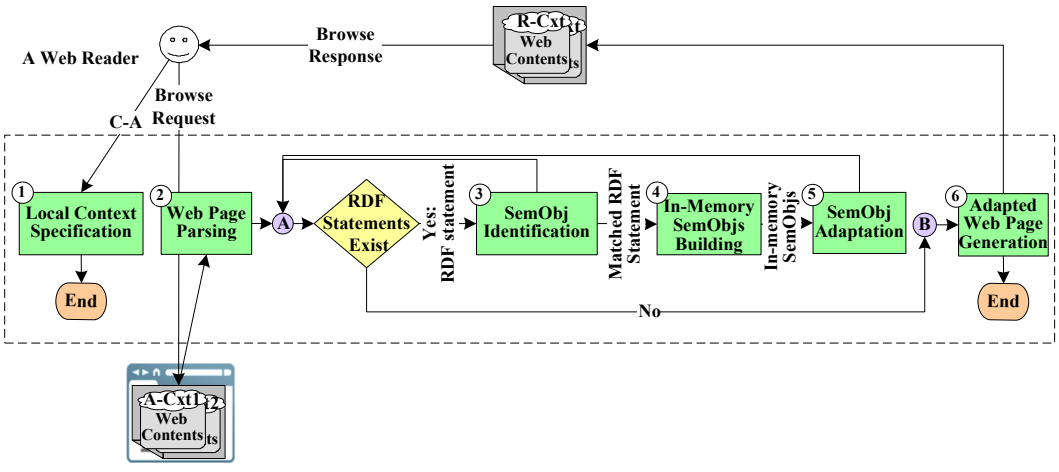

Figure 6.1: Overview of the adaptation process

6.4.1 Local Context Specification

Prior adapting semantic objects (i.e., pre-adaptation task), this task assists a Web reader to specify his local context information, and store the specified information in R-cxt document. In fact, this task is similar to the Task 1 involved in the Web annotation process described in Section 5.3.1. To keep this section self-contained, the following list paraphrases how a reader performs this task (see Figure 6.2[4]):

- A reader has to specify static context attributes at minimum. Also, he can specify one or more dynamic attributes if there is a specific need for unusual/specific dynamic values.
- Then, the specified values are stored in the R-Cxt document as instances of context attributes specified in the *LCO* (see Section 4.5).

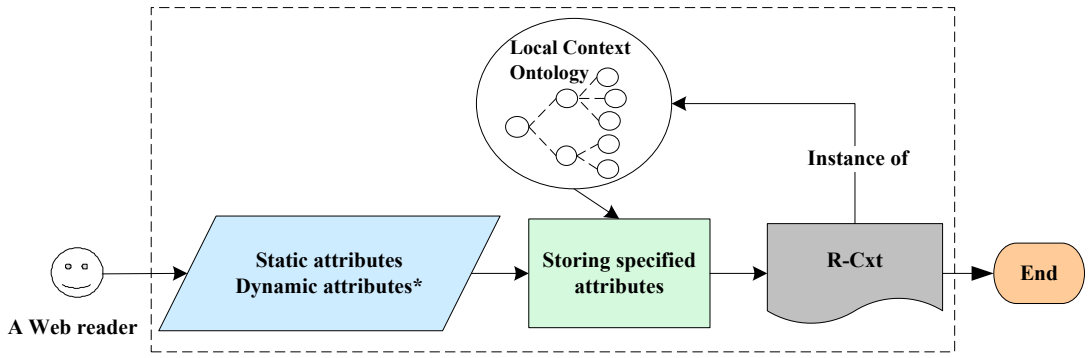

Figure 6.2: Adaptation process: local context specification flowchart

6.4.2 Web Page Parsing

Whenever a reader requests a Web page, this process analyzes the requested page in order to locate *CSCs* that are annotated as semantic objects as follows (See Figure 6.3):

- This task firstly builds an in-memory DOM[5] tree of requested page. Next, it sets up the context of Web page parsing based on the RDFa processing rules[6]. Afterwards, The DOM tree is recursively analyzed (i.e., from root element up to leaves) in the following steps until the end of its elements.

[4]The symbol * indicates that the specification of dynamic attributes is optional.
[5]Document Object Model
[6]More details about RDFa processing rules available on: `http://www.w3.org/TR/rdfa-syntax/`

- While the DOM tree has elements, the current DOM element is analyzed in order to detect a complete RDFa triple. If an RDFa triple detected, the next step is performed. Otherwise, the next DOM element in the DOM tree is analyzed.

- When it is detected, the RDFa triple is extracted and then mapped to an RDF statement in the form of subject-predicate-object. In general. the subject refers to the URI of the detected RDFa triple. This expresses a Web resource localized in the requested Web page. The object could be a Web content. In this case, this content expresses the physical representation of the Web resource. Finally, the predicate mostly consists of a set of RDFa attributes-values pairs. These pairs represent a set of attributes that describe the Web resource (i.e., semantic metadata). In our case, the subject refers to a $SemObj$ and the object the physical representation value V of a CSC. Finally, the predicate refers to a concept S and a minimum set of context attributes C that annotate this $SemObj$.

- When the elements of DOM tree are finished, the RDF statements that are extracted during this task are utilized as an input to the semantic objects identification task via connector (A).

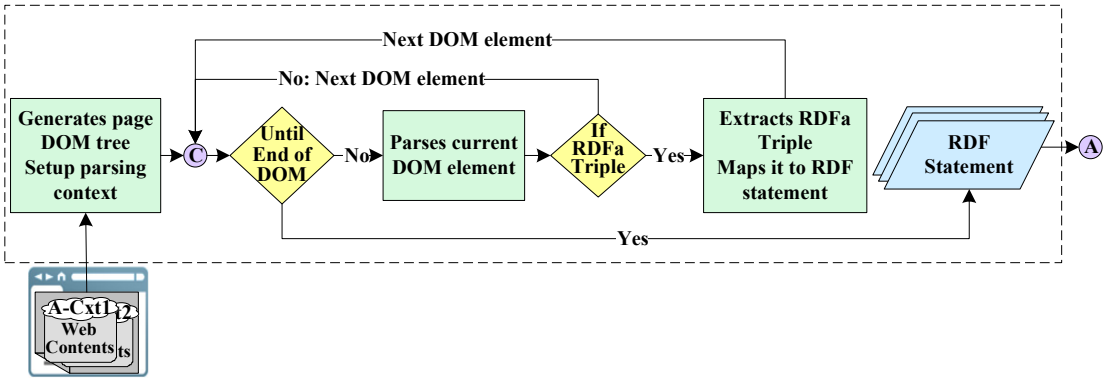

Figure 6.3: Adaptation process: Web Page Parsing flowchart

6.4.3 Semantic Objects Identification

Having parsed the requested page, this task attempts to identify each extracted RDF statement received by the previous task, via connector (A), as a semantic object as follows (see Figure 6.4):

- It initially attempts to match one of the RDF attribute represented in the predicate construct of the RDF statement under processing with one of the concept S. If the matching succeed, the next step is performed. Otherwise the next RDF statement is checked.

  Note that, it is assumed that authors' applications identify mappings from their concepts S to concepts identified in common ontologies. Also, the latter is adopted by readers' applications, as already discussed in Section 4.4.

- Based on the matched concept S, it communicates with the local context ontology LCO to extract the minimum set of context attributes. Then, it attempts to match them with context attributes corresponding to author's local context. If the matching succeed, the next step is performed, otherwise the next RDF statement is checked. Here, the minimum set of context attributes for each semantic object is also assumed common among users' applications.

- Based on the matched concept S, the RDF statement under processing is identified as a $SemObj$ of a particular type (e.g., date $SemObj$). Finally, it is utilized as an input to the in memory semantic objects building task.

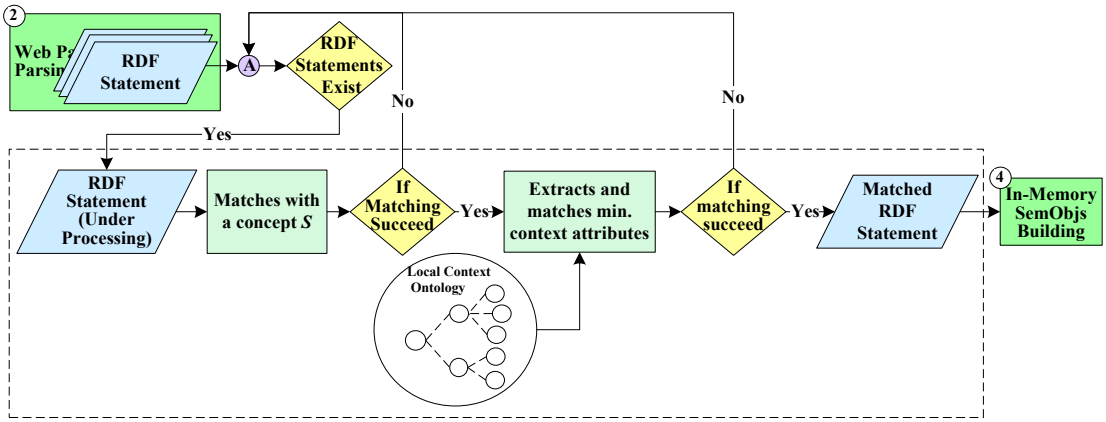

Figure 6.4: Adaptation process: Semantic Objects Identification flowchart

6.4.4 In Memory Semantic Objects Building

The role of this task is to prepare the environment of a reader's application in order to adapt the value of a CSC that is contained the matched RDF statement according to its reader's context. To this end, two in-memory $SemObj$ instances are created and utilized as input to the next task as follows (see Figure 6.5):

- Based on the concept S corresponding to the matched RDF statement, two instances of $SemObj$ are built in memory. Then, the information contained in this statement is assigned to one of them. This information identifies the concept S, the minimum set of context attributes and their values corresponding to an author's context C, and the value V of a CSC to be adapted.
- The concept S corresponding to the matched RDF statement is also assigned to the second $SemObj$ instance. Then, the minimum set of context attributes (i.e., C') corresponding to the second $SemObj$ instance are identified as follows:
  - It communicates with LCO to identify the relations between static and dynamic attributes corresponding to this type of $SemObj$.
  - Then, it extracts the values of static context attributes and the values of specified dynamic context attributes from the R-Cxt document. After that, the values of non specified dynamic attributes are inferred from the values of static attributes. Recall that, it is considered that this inference is based on the relations between static and dynamic attributes described in the LCO.
  - Finally, the values of the minimum context attributes corresponding to reader's context are assigned to the second $SemObj$ instance.
- Finally, the two $SemObj$ instances are utilized as input to the semantic object adaptation task.

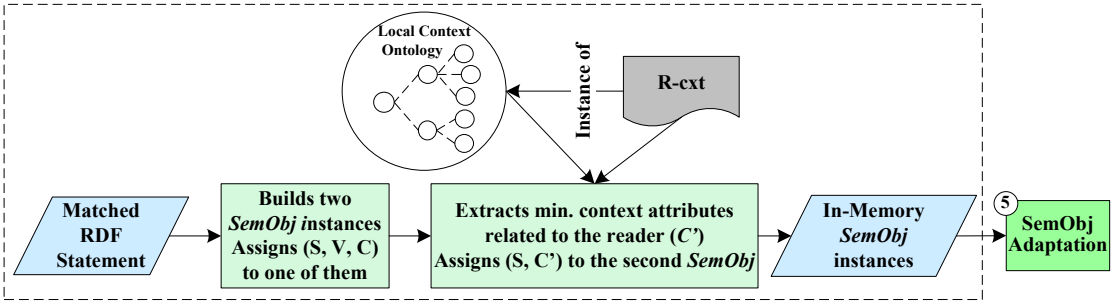

Figure 6.5: Adaptation process: In Memory Semantic Objects Building flowchart

6.4.5 Semantic Object Adaptation

Having built two $SemObj$ instances in memory, it is possible to adapt the value V of the first $SemObj$ instance from author's to reader's contexts. To do so, the adaptation

functions F dedicated to this type of $SemObj$ have to be located and applied. This is theoretically discussed in Section 6.2.1 above. Also, the following list paraphrases how this task accomplishes this (See Figure 6.6):

- The concept S is used to locate the set of adaptation functions F dedicated to the type of $SemObj$ instances that are built in memory. The adaptation functions for each type of $SemObj$ are described in details in Section 6.3.
- Each adaptation function f_i takes the value V and a subset of context attributes from both C and C' sets as parameters (illustrated as c, and c' in the flowchart). Then, it adapts V according to the reader's context V'.
- The value V' is assigned to the second $SemObj$ instance built in memory. The latter is semantically equivalent to the first $SemObj$ instance, but its value V' is represented according to the reader's context C', as already mentioned above.
- Finally, this task gives the control to the connector (A) in order to check if there are RDF statements extracted during Task 2 and can be identified as semantic objects.

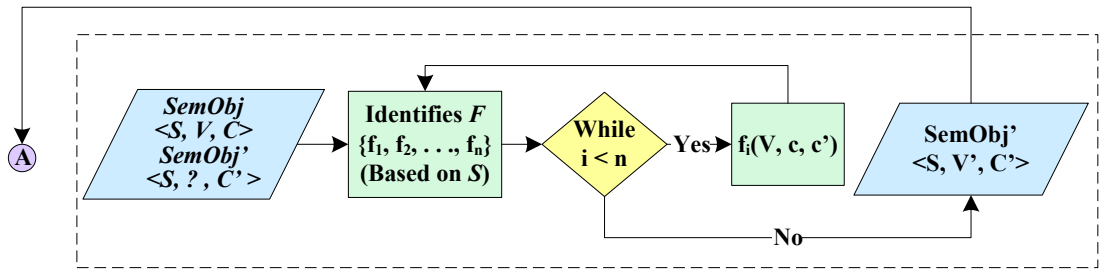

Figure 6.6: Adaptation process: Semantic Object Adaptation flowchart

6.4.6 Adapted Web Page Generation

When all RDF statements extracted during Tasks 2 are analyzed, the control is given to the connector (B). This connector starts this process in order to generate an adapted version from the requested Web page as follows (see Figure 6.7):

- For each matched $SemObj$ = <S, V, C>, this task locates the DOM element that represents the value V. Also, it locates the corresponding $Semobjs'$ = <S, V', C'> involved in the input. Next, the value V is appended with V'. In fact, the value

V has to be replaced by its corresponding V'. However, we deem appropriate to keep the original value and add the adapted value after it, between brackets, in order to ensure a good reader's understanding.

- Finally, the adapted version of the requested page is presented according to the reader's context.

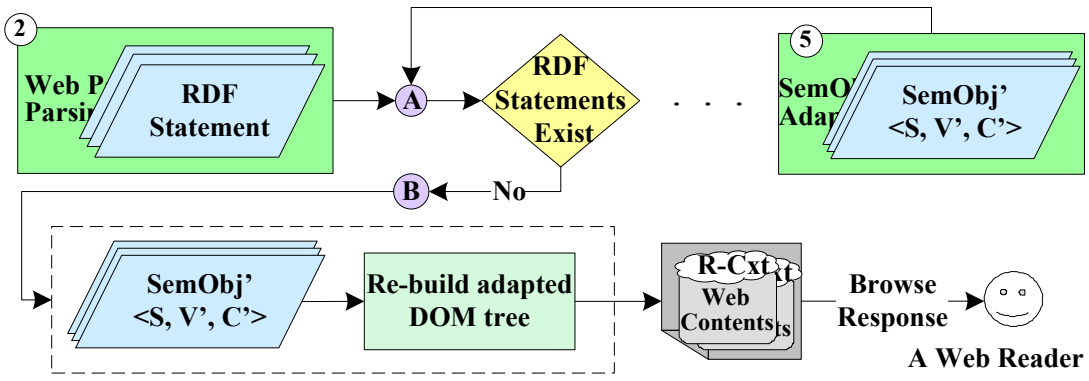

Figure 6.7: Adaptation process: Adapted Web Page Generation flowchart

6.5 Adaptation Engine: Architecture and Prototype

With respect to the architecture perspective, Section 4.7 introduces an adaptation engine as an extension to traditional Web browsers. The role of this engine is to embody the aforementioned adaptation process. This section describes the internal structure of this engine. Also, it presents a prototype which proves the validity of the proposed adaptation process.

6.5.1 Internal Structure

Figure 6.8 depicts two views of the adaptation engine: process and architecture views. The *process view* is already detailed in the previous section. The architecture view encompasses two modules: interface and adaptation functions modules. In addition, it utilizes the concept description layer that is described in the general architecture (See Section 4.7).

The *interface* module extends the interface of traditional Web browsers with the L-C form. This form allows a reader to specify his local context information. The *adaptation functions* module embodies the aforementioned adaptation tasks, where each adaptation

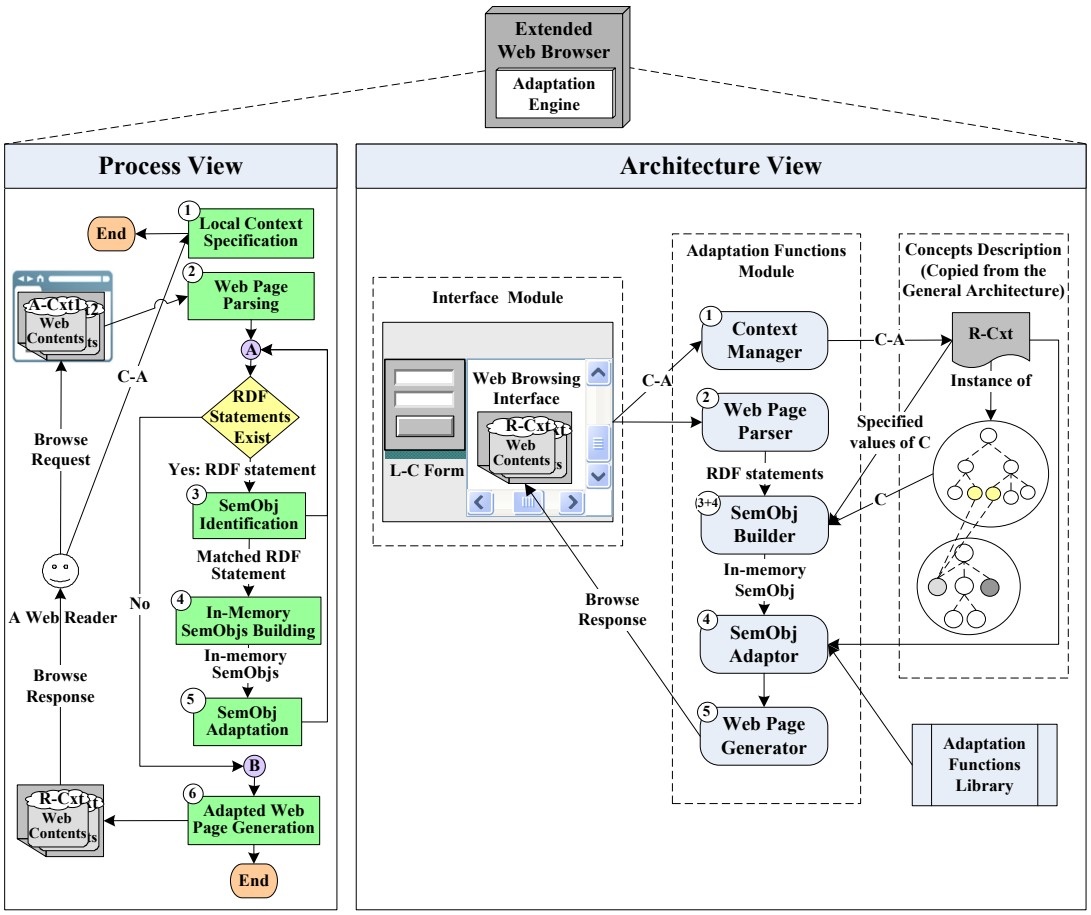

Figure 6.8: Adaptation engine: internal architecture view

task shown in the process view is designed as a function in this module[7]. These functions are invoked upon readers' interactions with the interface components. For instance, when a Web reader requests a Web page, the Web page parser generates the DOM tree of this page and analyzes it in order to extract RDF statements involved in it (if any). Afterwards, it invokes the *SemObj* builder function to identify the extracted statements as *SemObjs*, and builds two in-memory instances for each identified *SemObj*.

In addition, The architecture view involves a set of pre-defined adaptation functions. These functions are stored in the *adaptation functions library* and embody the set of adaptation functions that are dedicated to adapt *SemObjs* (see Section 6.3). These pre-defined functions are invoked by the *SemObj* adaptor function in order to adapt the

[7]In Figure 6.8, the same numbers are given to the adaptation functions and their corresponding adaptation tasks.

identified *SemObjs*.

6.5.2 Web Adaptation Prototype

In addition to the aforementioned architecture, we propose an annotation engine prototype as a proof-of-concept. Our prototype is implemented as a FireFox browser extension under Eclipse™ environment. The L-C form is implemented using an XML-based user interface Language known as XUL[8]. Also, we utilize javascript™APIs to implement the functions involved in the adaptation module.

With respect to the adaptation functions library, we benefit from the rich java™APIs for implementing them. In addition, a technology called XPCOM[9] is utilized in order to allow the *SemObj* adaptor function to interact with the adaptation functions library. Basically, this technology provides an interface between java and javascript APIs which are used in the implementation of Firefox extensions.

Figure 6.9 presents a screenshot of our prototype and illustrates the adaptation of different types of *SemObjs* according to a French reader's local context.

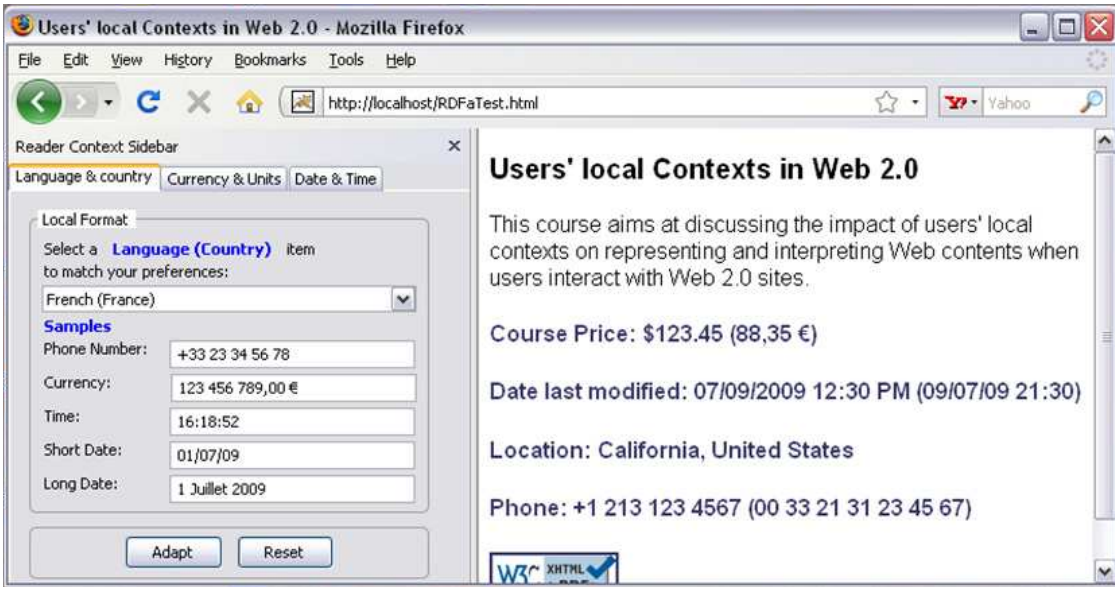

Figure 6.9: A screenshot of the extended FireFox Browser

[8] `https://developer.mozilla.org/en/xul`
[9] Cross Platform Component Object Model: `https://developer.mozilla.org/en/xpcom`.

Part III

Web 2.0 Usability Evaluation

Chapter 7

Web 2.0 Usability Evaluation Methodology

7.1 Introduction

The utilization of one or more usability improvement means are considered not sufficient for ensuring the usability of users' interactions with a Web site. Even though accurate improvement means are utilized, it is still necessary to evaluate the results of users' interactions against the desired level of usability (i.e., usability criteria). Therefore, the usability evaluation is essential and principal for validating the usefulness of the utilized usability improvement means [79].

A successful usability evaluation should provide a way to assess users' interactions with the intended Web site(s) during the usability design stage (see usability engineering stages in Section 1.1). This helps to void expensive and complex re-design steps related to the evaluated interactions. In addition, it should provide a way for validating if real users can easily interact with the intended Web site(s) and perform tasks they want. This helps to detect unexpected usability problems and also the actual reason of these problems, as they are encountered by real users and during their actual use. Finally, it should provide a way to get feedbacks from users about their satisfaction, whereby these feedbacks are considered useful inputs for subsequent re-design steps.

From technical viewpoint, there are two main usability evaluation approaches. The first approach is called *usability inspection* and it utilizes a number of usability principles and guidelines to inspect several usability aspects during design stage. The second one is called *user-testing* and it is used to evaluate whether the actual results achieved by users comply with the desired usability level. However, a wide range of usability evaluation methods are used to apply each of these approach. One method may efficiently work in an application domain, but it is useless in other domains. For instance, a method called *controlled experiment* is recommended to evaluate the actual results of Web adaptation approaches, but it is useless (or not recommended) to evaluate users' interactions when

they perform many interactive tasks (see Section 2.4). In this context, most usability experts recommend to use more than one evaluation methods and to select the best methods that comply with an application domain to conduct usability evaluation [57].

With respect to our approach, The term *Web 2.0 usability* is defined in Section 1.1 in terms of effectiveness, efficiency, and satisfaction criteria that characterize users' interactions with Web 2.0 pages. Accordingly, a number of usability problems related to the representation and interpretation of $CSCs$ are specified in Section 1.2.2. Afterwards, Web annotation and Web adaptation are introduced as Web usability improvement means to handle these problems. Finally, The term *Web 2.0 usability evaluation* is introduced in Section 1.4 to address two issues: first, inspecting several usability aspects during the design of Web annotation and Web adaptation; secondly, introducing a methodology to evaluate the actual results of users' interactions after applying these usability improvement means.

This chapter aims at discussing these two issues in more details. Prior this, it is necessary to discuss the following related aspects. Our intention is to avoid any misunderstanding that readers might encounter, and also to specify the boundary of our proposed evaluation methodology.

First, we can consider that the Web 2.0 usability analysis that is discussed in Section 1.2 and detailed in Chapter 3 is an initial evaluation step of Web 2.0 usability, and the results of applying the following evaluation methodology is the second evaluation step (also called intermediary evaluation). Our consideration comes from the fact that there is no sharp distinction between the usability analysis, design, and evaluation phases that are mentioned in Section 1.1. For example, usability analysis phase can be considered as an evaluation of the current state of users' interaction, and usability evaluation phase can be considered as evaluation of users' interactions after applying one or more usability improvements means. Moreover, several evaluation activities are usually conducted at design phase, as we will see in Section 7.2. Our consideration is also compatible with the most usability engineering methodologies, which consider the evaluation of Web usability is an iterative task [46, 79, 85].

Secondly, the proposed usability evaluation methodology details our recommendation on how to evaluate usability aspects related to Web annotation and Web adaptation. The design of the latter means as extensions to users' applications implies that there are other usability aspects related to these applications. Our evaluation methodology does not consider these aspects. More specifically, our goal is to evaluate users' interactions when they annotate $CSCs$ at creation/update time and the adaptation of annotated $CSCs$ at browsing time. However, it is not intended to evaluate the usability aspects

related to the creation, update, and browsing of Web contents. In addition, the user interfaces that are utilized for these purposes are also not considered.

Thirdly, usability aspects are strongly interrelated in such a way that one usability aspect affects on and/or is affected by other usability aspects. Hence, usability evaluation have to consider all usability aspects in order to ensure a successful evaluation [57, 90]. In this sense, there are several reasons that restrict us to conduct the proposed usability evaluation methodology in practice (at least currently). The following mention the main two reasons:

1. As usability aspects are strongly interrelated, the usability evaluation of authors' interactions has to consider usability aspects that are related to the creation, update, and annotation of Web contents all together. However, Web 2.0 is extremely new domain and a few approaches target the usability aspects corresponding to this kind of interaction (i.e., users/Web 2.0 sites interactions) [86, 93]. As a result, most usability issues are still not addressed yet, and addressing them require more time and research efforts.

2. Usability aspects related to readers' interactions are also strongly interrelated to usability aspects related to authors' interactions. For example, Web adaptation engine can adapt $CSCs$ if the later are annotated by their authors at creation and update time correctly. Therefore, the evaluation of readers' interactions is considered interesting if and only if $CSCs$ are annotated completely and correctly.

As a result, we deem appropriate to propose a methodology that details our recommendation on how to evaluate usability aspects related to Web annotation and Web adaptation. Then, giving the responsibility for developers (e.g., Web 2.0 site developer or usability evaluators) who adopt our approach to integrate this methodology with their own traditional Web usability evaluation approaches. Also, relying on them to prepare the needed materials for conducting this methodology in practice.

The rest of this chapter is structured as follows. Section 7.2 discusses several usability inspection steps that are made during the design phase. Next, the proposed usability evaluation methodology related to Web annotation and Web adaptation is described in Section 7.3.

7.2 Web 2.0 Usability Inspection During Design Phase

During the Web 2.0 usability design phase (i.e., Chapters 4 - 6), an extensive attention is given to the ways users actually interact with Web 2.0 sites. Our intention is to optimize

the representation and the interpretation of $CSCs$ according to how users need and/or prefer to do this, rather than forcing them to change the way they represent and interpret these $CSCs$. This intention is mainly derived from the User-Centered Design (UCD). UCD refers to a design philosophy that considers users' needs and interests as the key points of Web sites design and implementation. Their major characteristics are the active involvement of users, clear understanding of tasks that users have to perform, and an iterative design of a Web site based on users' feedbacks. Based on these characteristics, a number of design principles and guidelines have been proposed and utilized in order to facilitate the design of usable interfaces and usable interactions [87, 90].

Part of our attention includes inspections of usability issues related to several design steps. These inspections were discussed when they were conducted and a number of design decisions were taken according to them. Here, we aim at emphasizing on these inspections and present them together from usability evaluation perspective. The following list paraphrases these assessments and referring to the design decisions that were taken accordingly:

1. *Local context specification*

   The first usability issue is related to the specification of local context attributes. In practice, Web users could face difficulties to specify the values of some context attributes that they use to represent $CSCs$.

   This issue is considered during the representation of $CSCs$ as semantic objects and during the design of the local context ontology (LCO) as follows (see Chapter 4). First, these context attributes are classified as dynamic attributes, and they are enriched with one or more context attributes. Secondly, context attributes that are easy to specify their values are classified as static context attributes. The values of latter attributes are used to determine the values the dynamic attributes corresponding to them. Finally, according to this classification, the LCO is designed around two main concepts: country and community conventions. Then, these conventions are related to their origins (e.g., country and language), whereby the values of conventions are determined from the values of their origins. Our design is based on assumption that users can specify the values of the origins easily.

2. *Web annotation*

   During the design of Web annotation, three usability issues are inspected. The first two issues are related to Web annotation process. In fact, authors are mostly non-experts. Thus, they do not know the relations between $CSCs$ and context

information used to represent them. Also, Web annotation technology (i.e., the RDFa) is extremely new technology, and using it by authors to manually annotate $CSCs$ is too difficult and prone to serious errors.

The design decision to address these two issues is to assist authors for interactively annotating $CSCs$ as easy as formatting text in traditional word editors. In addition, underlining annotated CSC and providing a suitable warning message in case there is an error in the annotation (see Section 5.3).

The third issue is related to the semantic visibility of context attributes. More specifically, we deem appropriate to infer the values of dynamic attributes from corresponding static attributes at annotation time, and to annotate each CSC with a minimum set of context attributes corresponding to it as a $SemObj$. This helps a reader to interpret annotated $CSCs$ correctly even though it is not possible to adapt them according to his/her local context, as context attributes can be made visible. Consequently, this issue is considered in the evaluation of Web annotation alternatives (see Section 5.2).

3. *User interface of Web annotation and adaptation engines*

   Several usability guidelines are followed during the design of the Web annotation and Web adaptation engines. First, an example is given in the local context panel in order to assist users (authors and readers) to specify their minimum local context attributes. Also, users can open an extended panel in case they need to specify specific values for one or more dynamic attributes. Secondly, in case a Web author annotate a CSC incorrectly, this CSC is underlined and a suitable warning message is provided to assist this author to correct the encountered error. Thirdly, when Web readers adapt $CSCs$ involved in a Web page, the original versions of the annotated $CSCs$ are preserved, and the adapted versions are added after the original versions. This help readers to better understand these $CSCs$.

7.3 Web 2.0 User-Testing Evaluation

The inspection of usability aspects during the design phase is an important step towards improving the Web 2.0 usability corresponding to authors' and readers' interactions. However, the inspection by itself does not guarantee meeting the desired level of Web 2.0 usability, as aforementioned. Therefore, still there is a need to validate this with actual users' interactions.

As already discussed in Section 2.4, user-testing evaluation could target objective

aspects such as task performance (e.g., execution time) and numbers of errors encountered. Also, it could target subjective aspects such as users' satisfactions. In addition, several methodologies has been proposed for planning and managing tasks that should be performed during this evaluation. Finally, several methods are used to collect and/or analyze users' data corresponding to each task specified the evaluation methodology [46, 79, 85].

This section proposes a user-testing evaluation methodology that is derived from the existing evaluation methodologies and aims at evaluating the interactions of authors and readers. Our methodology starts by defining a number of measurable usability factors from the desired usability level introduced in Section 1.2.2. Indeed, the desired usability level is identified as a number of usability criteria related to both authors and readers. These criteria are qualitative in nature, and thus it is difficult to compare them with the actual result of users' interaction. In contrast, these criteria are directly affected (or can be determined) by a number of measurable factors[1]. For example, the effectiveness criterion can be related to the ability of users to perform a task and the number of errors encountered during the accomplishment of this task [114].

Next, our methodology identifies three types of users-testing evaluations. Each of these evaluations utilizes one or more usability evaluation methods and recruits a number of users (called representative users). The former is used to collect and analyze representative users' data related the identified measurable factors as follows:

1. *Effectiveness and efficiency of Web annotation.* A scenario-based evaluation method is utilized to evaluate the effectiveness and the efficiency of authors' interactions when they annotate $CSCs$ with their corresponding local context information.

2. *Effectiveness and efficiency of Web adaptation.* A combination of scenario-based and controlled experiment evaluation methods are utilized in order to evaluate the effectiveness and the efficiency of readers' interactions when they interpret $CSCs$ before and after adapting them according to their local contexts.

3. *Evaluation of Users' satisfaction.* After participating in one of the two user-testing evaluation, each representative user is asked to provide feedbacks related to his satisfaction about the usability of the process under evaluation (i.e., Web annotation or Web adaptation process). To conduct this type of evaluation, we recommend to utilize a Web-based questionaire method.

[1] In [114], these factors are called usage indicators.

7.3.1 Author-Testing Evaluation

Section 1.2.2 defines the desired usability level related to Web authors' interactions in terms of three usability criteria. These criteria can be paraphrased as follows. First, the interactions of Web authors are considered *effective* when they are able to annotate all types of $CSCs$ with their corresponding local context information in an accurate manner, so that readers' applications are able to interpret and adapt the annotated $CSCs$. Secondly, their interactions (i.e., authors) are considered *efficient* when the efforts (i.e., time) needed to annotate $CSCs$ effectively are relatively not too high. Thirdly, Web authors are considered *satisfied* when they do not face difficulties to specify their local context and to annotate all types of $CSCs$, such that they do not reject to annotate new $CSCs$ in the future. Subsequently, Chapter 4 discusses several annotation alternatives and presents an interactive annotation process in order to achieve the aforementioned desired usability level.

In order to evaluate the actual interaction of authors, a number of measurable factors that affect on the above usability criteria are defined. Then, one or more questions are formulated for each measurable factor. The answers of these questions represent the actual results/feedbacks achieved by representative authors who participate in these evaluations. Figure 7.1 presents the usability criteria and their corresponding measurable factors and questions.

Measurable Factors and Questions

The effectiveness criterion is divided into two sub-criteria: completeness and accuracy. Two measurable factors are defined for the completeness criterion. The first factor concerns the ability of authors to specify their local context information (static attributes at minimum). The second one concerns the ability of authors to annotate $CSCs$ exist in Web contents to be evaluated. Consequently, the following evaluation questions are formulated in order to measure these factors:

- How many representative authors specify their local context information (at least static attributes)?
- Are authors able to specify the values of one or more dynamic attributes, in case they need specific values for these attributes (e.g., sales tax rate and date format)?
- How many $CSCs$ each representative author annotates compared with the overall $CSCs$ he creates or updates?

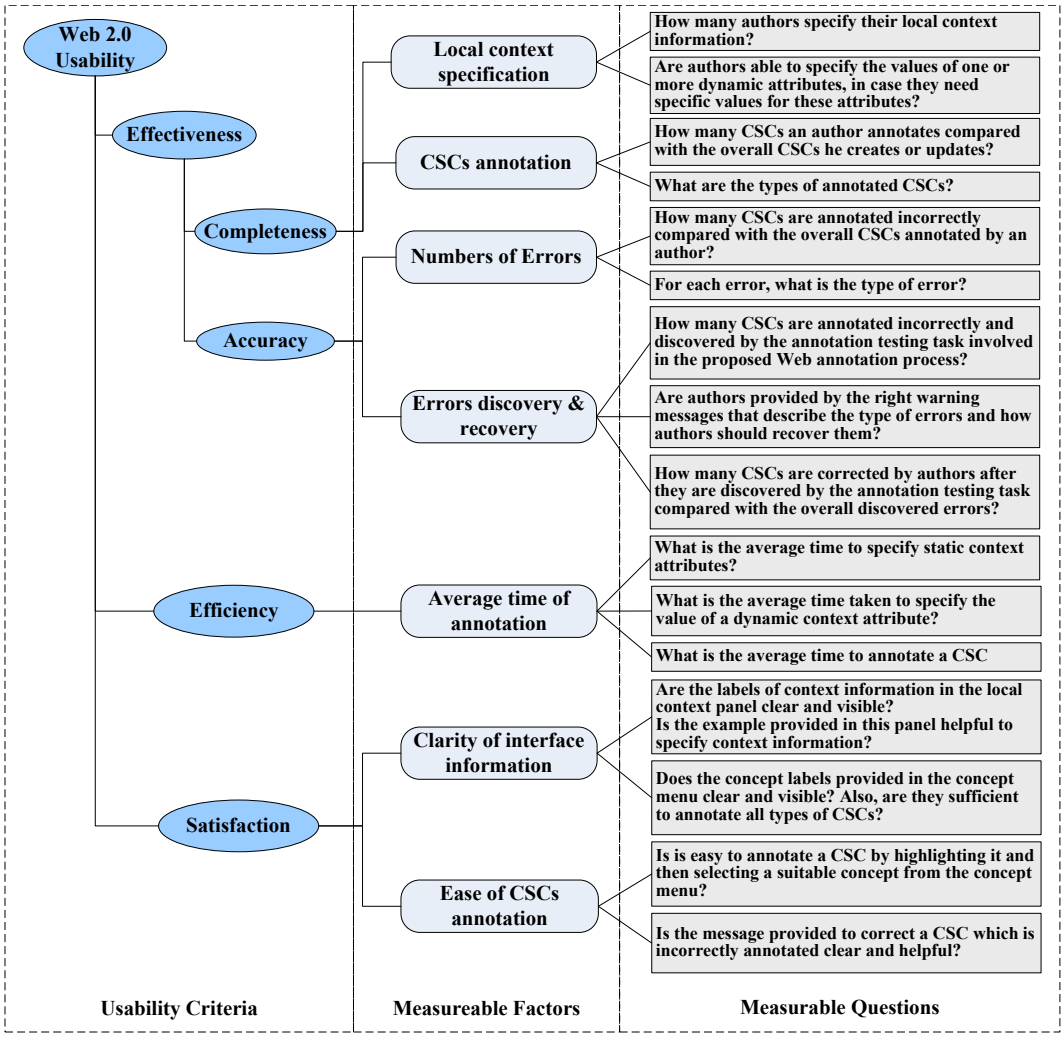

Figure 7.1: Measurable factors and questions for evaluating the Web 2.0 usability of authors

- What are the types of annotated $CSCs$?

In similar way, two measurable factors are defined and related to the accuracy criterion. The first factor concerns the number of errors encountered by authors during the annotation process. The second one concerns the discovery and the recovery of the encountered errors by the annotation process. To measure these factors, the following questions are formulated:

- How many $CSCs$ are annotated incorrectly compared with the overall $CSCs$ annotated by each representative author?

- For each error, what is the type of error? For instance, is a CSC annotated by incorrect semantic concept S, or does an annotated CSC not comply with context information specified by the representative author?
- How many $CSCs$ are annotated incorrectly and discovered by the annotation testing task involved in the Web annotation process (see Section 5.3)?
- Are authors provided by the right warning messages that describe the type of errors and how authors should recover them?
- How many $CSCs$ are corrected by authors after they are discovered by the annotation testing task compared with the overall discovered errors?

The efficiency criterion is usually related to the time taken to accomplish the task(s) to be evaluated. In this sense, the efficiency of the annotation process is related to the time taken from an author to specify his context and to annotate $CSCs$. This time is closely interrelated with the time taken to create and/or update Web contents. Hence, the efficiency evaluation of the annotation process is considered interesting only if it is addressed together with the efficiency of Web contents creation and update tasks. One way to simplify this is to ask representative authors to annotate $CSCs$ after they finish creating and/or updating Web contents. Then, the time taken for annotating a CSC can be measured by dividing the time taken to annotate all $CSCs$ on the numbers of annotated $CSCs$. To sum up, the following questions are formulated to evaluate the efficiency of the annotation process:

- What is the average time to specify static context attributes?
- What is the time taken to specify the values of one or more dynamic context attributes in case a representative author needs specific values for these attributes?
- What is the average time to annotate a CSC?

The satisfaction criterion is more oriented to subjective attitude of authors, as already mentioned in Section 1.2.2. For instance, the satisfaction of authors who use the extended Web editor proposed in Section 5.4.2 can be related to two factors. The first one concerns the clarity of information involved the Web annotation interfaces (i.e., local context panel and concept menu). The second factor concerns the ease of annotating $CSCs$. These two factors can be evaluated by asking authors the following questions:

- Is the context information provided in the local context panel clear and visible? For example, are the example and the labels of context information provided in this panel helpful to specify context information?
- Are the concept labels provided in the concept menu clear and visible? Also, are they sufficient to annotate all types of $CSCs$?
- Is it easy to annotate a CSC by highlighting it and then selecting a suitable concept from the concept menu?
- Is the message provided to correct a CSC which is incorrectly annotated clear and helpful?

Effectiveness and Efficiency of Web Annotation

Our recommendation is to utilize a scenario-based evaluation method to answer the questions related to the effectiveness and efficiency criteria formulated above. Recall that, a number of predefined scenarios designed in this method in order to cover the major functionalities that are intended to be evaluated. Afterwards, a number of representative users are asked to perform these scenarios. During this, users' data related to usability aspects are collected in order to be compared with the desired usability level (See Section 2.4).

Reflecting this on our Web annotation evaluation, we have to select a number of representative authors in order to participate in the evaluation. To this end, five local communities and six authors from each community who have good experience to deel with the Web 2.0 are deemed suitable to represent the entire population of authors. These authors are distributed into three groups, where each group consists of two authors from each local community. As a result, each group has ten representative authors who are originated from different five communities.

The members of each group are asked to use the extended Web editor (see Section 5.4.2) in order to specify their context information and to carry out one of the following annotation scenario:

1. *Contents creation and annotation.* In this scenario, representative authors are asked to create Web contents that have a number of $CSCs$ of different types. Next, they are asked to interactively annotate these contents with suitable concepts S, and to correct the annotated $CSCs$ that are incorrectly annotated before publishing the created contents, as already discussed in Section 5.3.

2. *Contents update and annotation.* Each representative author is asked to browse Web contents that are created by other authors in this scenario. Next, he is asked to interactively update and annotate a number of $CSCs$ which are already annotated by their original authors, and to correcte the $CSCs$ that are incorrectly annotated (if any).

3. *Contents annotation.* In this scenario, representative authors are asked to annotate a number $CSCs$ after they finish creating/updating Web contents.

The role of the first and second scenarios is to evaluate the correctness and the accuracy of annotating $CSCs$ created and updated by representative authors. Indeed, the first scenario evaluates the annotation of $CSCs$ that are created by the same representative author. The second one evaluates the annotation of $CSCs$ that created and updated by many authors. Our goal is to evaluate all possible Web 2.0 scenarios that can be performed by authors (see Section 3.2). The role of the third scenario is evaluate the efficiency criterion (i.e., the time spent to annotate a CSC). In other word, this scenario is used to calculate the time that is spent to annotate $CSCs$ involved in the Web contents.

In practice, each representative author is recruited via sending him an email. This email includes a link to a "welcome" page explaining the intended scenario he have to carry out. After that, he is redirected to the evaluation page to start carrying out this scenario. During this, the actual data related to the questions formulated above have to be collected.

With respect to the effectiveness criterion, part of this data can be collected from the annotated $CSCs$ such as the number of $CSCs$ that are annotated by a representative author compared with the overall $CSCs$ and whether the annotated $CSCs$ are correctly annotated. For other types of data such as the number and the type of $CSCs$ that are annotated incorrectly and whether they are corrected or not, we recommend to utilize the capabilities of javascript in order to store these data in a cookie document. With respect to the efficiency criterion, we also recommend to utilize the capabilities of javascript to start a session and allow the representative author to pause, continue, and finish this session. When it is finished, the session time is also stored in a cookie document. Finally, the collected data are then returned, tabulated, and analyzed in order to discover any potential usability problems.

Evaluation of Authors' Satisfaction

Having finished conducting one of the above scenario, each participated author is also asked to answer the questions related to satisfaction criterion formulated above. To this end, we recommend to utilize a Web-based questionaire method as follows. The questions involved in the questionaire are designed using a Web form. Also, the answer of each question could be predefined using 5-point Likert scale[2], or could be open-ended answer where authors are asked to give their feedbacks, or both. In addition, the questions proposed above can be paraphrased, divided into sub-questions, and/or one question can be asked in two different ways. This help authors to answer these questions. Also, it helps evaluators to check whether authors seriously answer these questions or not.

In practice, each participated author is redirected to the questionaire Web form after finishing the scenario he is recruited to carry out. Then, he is asked to answer questions involved in this questionaire and submit the answers after he finish. Finally, the answers are then stored for latter analysis by evaluators.

7.3.2 Reader-Testing Evaluation

Section 1.2.2 also defines the desired usability level related to Web readers' interactions in terms of three usability criteria. These criteria can be paraphrased as follows. The interactions of Web readers are considered effective and efficient when they are able to specify their local contexts and interpret all $CSCs$ involved in a Web page accordingly. This implies the following. Readers do not misunderstand $CSCs$ that are created and updated by authors who have different local contexts. Also, the additional efforts required to interpret these $CSCs$ are no longer required, as they are presented according to their readers' contexts. In addition, readers are considered satisfied if they do not face difficulties to specify their local context and if they interpret $CSCs$ according to their local contexts. Subsequently, Chapter 6 proposes an adaptation process in order to achieve the aforementioned desired usability level.

In order to evaluate the actual interaction of readers, a number of measurable factors are defined and one or more questions are formulated for each measurable factor like it is made in the above author-testing evaluation. Figure 7.1 presents the usability criteria and their corresponding measurable factors and questions.

[2] A Likert scale is a type of question in which users are asked to evaluate the level of their agreements in terms of five scales: strongly disagree, disagree, fair, agree, strongly agree

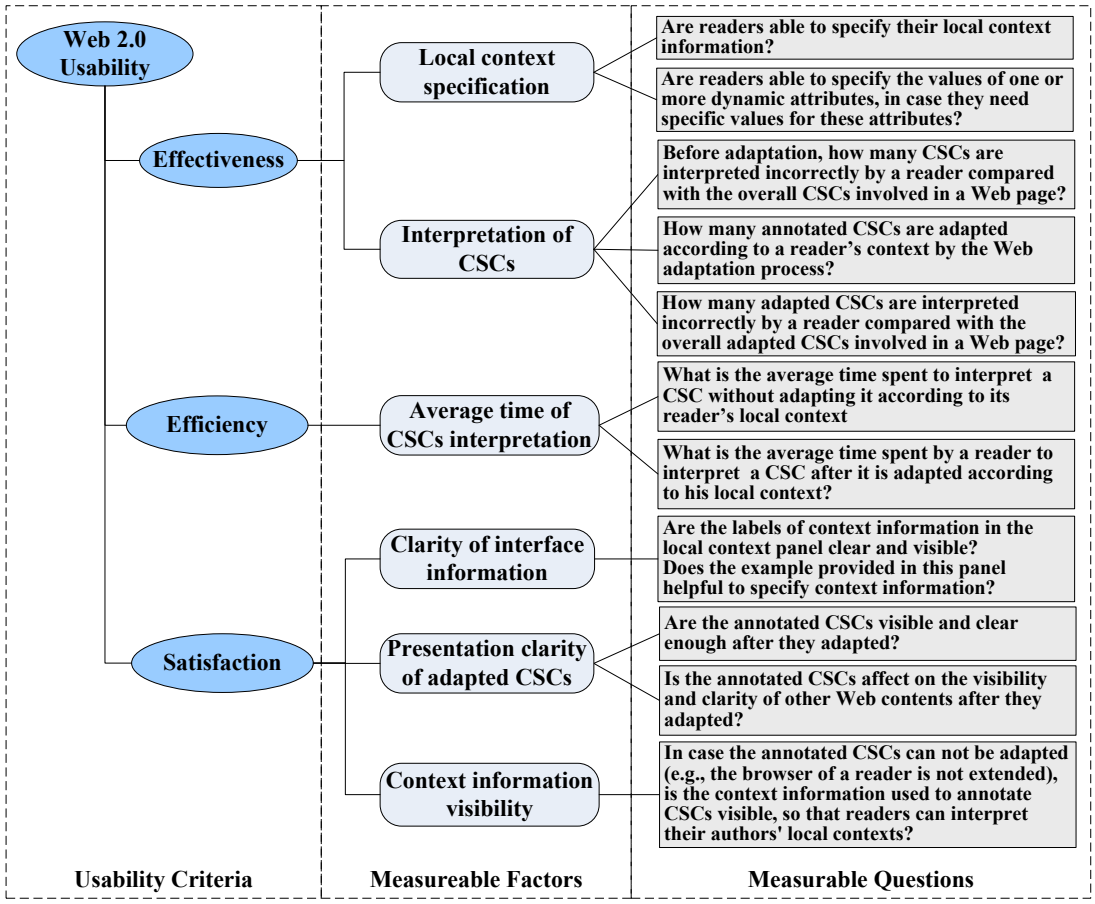

Figure 7.2: Measurable factors and questions for evaluating the Web 2.0 usability of readers

Measurable Factors and Questions

Two measurable factors are defined and related to the effectiveness criterion. The first factor concerns the ability of readers to specify their contexts information correctly and accurately. To consider this factor, the same questions concerning the ability of authors to specify context information are also utilized here. The second factor concerns the interpretation of $CSCs$ that were annotated by authors before and after adapting them according to their reader's context. Hence, the following questions are formulated to evaluate this factor:

- Before they are adapted according to their reader's context, how many $CSCs$ are interpreted incorrectly by this reader compared with the overall $CSCs$ involved in a Web page?

- How many annotated *CSCs* involved in a Web page are adapted according to their readers context by the Web adaptation process?
- After they are adapted according to their reader's context, how many *CSCs* are interpreted incorrectly by this reader compared with the overall adapted *CSCs* involved in a Web page?

With respect to the efficiency criterion, it is related to the time teken to browse and interpret Web contents that consist of several *CSCs*. In this sense, browsing time have to be decreased after *CSCs* are adapted according to their reader's local context (i.e., no additional time required to interpret them). This time can be evaluated by formulating two questions concerning the time spent to browse Web contents before and after adaptation.

The satisfaction criterion is related to subjective attitude of readers. Hence, the satisfaction of readers who use the extended Web browser proposed in Section 6.5.2 can be related to three factors. The first one concerns the the clarity of information involved in the Web adaptation interface (i.e., local context panel). This can be evaluated by utilizing the same questions that are used to evaluate the clarity of local context panel of the extended Web editor. The second factor concerns the clarity of *CSCs* presentation after they are adapted. The third factor concerns the visibility of annotation information in case the annotated *CSCs* can not be adapted according to reader's context (e.g., the browser of a reader is not extended). The last two factors can be evaluated by asking authors the following questions:

- Are the annotated *CSCs* visible and clear enough after they are adapted?
- Is the annotated *CSCs* affect the visibility and clarity of other Web contents after they are adapted?
- In case the annotated *CSCs* can not be adapted, is the context information used to annotate *CSCs* visible, so that readers can interpret their authors' contexts information?

Effectiveness and Efficiency of Web Adaptation

In order to answer the questions related to the effectiveness and the efficiency of readers' interactions, we recommend to utilize a combination of scenario-based evaluation and controlled experiment[3] methods. Also, our recommendation is to recruit a number of

[3]As discussed in Section 2.4, the main idea of the controlled experiment method is to evaluate the consequences of one or more usability hypotheses on the actual users' interactions.

representative readers in order to browse Web contents that were created/updated by the representative Web authors who carried out the above author-testing evaluation. More specifically, two groups of readers, each of which has ten representative readers from different five local communities are deemed suitable to represent the entire population of Web readers. There readers are asked to browse Web contents that were created by a single author and/or created and updated by multiple authors as described in the following scenarios:

1. *Web contents browsing without adaptation.* In this scenario, the members of the first group are asked to browse the Web contents that were created/updated by representative authors *without* adapting the annotated $CSCs$ according to their local contexts.

2. *Web contents browsing after adaptation.* The members of the second group are asked to extend their Web browser with adaptation engine proposed in Section 6. Afterwards, they asked to specify their context information and browse the Web contents *after* adapting the annotated $CSCs$ involved in these contents.

During conducting these two scenarios, the actual data that are needed to answer the above questions are collected from representative readers. For example, data corresponding to the numbers of $CSCs$ that are interpreted incorrectly and the time taken to interpret them before and after their adaptation are collected. Afterwards, the controlled experiment method is utilized to analyze these data by comparing the actual results before and after the adaptation of $CSCs$.

Practically, an email is sent to each representative reader, like in the above author-testing evaluation. This email includes a link to a "welcome" page explaining the intended scenario he has to carry out. After that, he is redirected to the evaluation page to start carrying out this scenario. During this, the actual data related to the questions formulated above have to be collected. Here, we also recommend to utilize the capabilities of javascript in order to collect and store this data. Then, the collected data are tabulated and analyzed in order to discover any potential usability problems.

Evaluation of Readers' Satisfaction

Having finished conducting the second scenario, the members of the second readers' group are asked to answer the questions related to satisfaction criterion formulated above using a Web-based questionaire method. Like the evaluation of authors satisfaction, the questions proposed above can be paraphrased, divided into sub-questions, and/or one

question can be asked in two different ways. Also, the answer of the questions involved this questionaire could be of type 5-point Likert scale, open-ended, or both. Finally, the answers of the questions are stored for analysis after the participated readers submit the questionaire form.

Part IV

Web 2.0 Usability: Conclusion

Chapter 8

Summary and Conclusions

8.1 Summary

In this thesis, we have focused on handling the local contexts of Web users during their interactions with Web 2.0 sites. We have discussed several discrepancies that arise when Web users browse several types of Web contents (i.e., Context-Sensitive Contents, or $CSCs$). These discrepancies are arisen since Web users belong to different local communities and they implicitly use their local contexts to represent and interpret $CSCs$. In addition, we have illustrated that these discrepancies increase when Web users interact with Web 2.0 sites for two reasons. First, Web contents could be created and updated from different authors who have different local contexts. Secondly, Web contents from different Web sites could be aggregated and displayed together in a single Web page. Hence, Web users mostly browse the contents of a single Web page that are represented according to different local contexts.

From usability point of view, the aforementioned issues hamper the interactions of Web users with Web 2.0 sites. In this sense, the term Web 2.0 usability has been utilized to refer to the effectiveness (completeness and accuracy), efficiency, and satisfaction criteria that characterize the interactions of users with Web 2.0 sites when they represent and interpret $CSCs$. Afterwards, the following usability problems have been defined. First, Web users require additional efforts to interpret $CSCs$ or even they could misinterpret their semantics, as these $CSCs$ are represented according to their authors' local contexts (inefficiency and inaccuracy problems, respectively). Secondly, the representation of $CSCs$ are *incomplete* as the authors' local contexts used to represent them are not explicitly specified. Hence, it is not possible to adapt these $CSCs$ according their readers' local contexts.

The terms *Web annotation and Web adaptation* have been defined as usability improvements means to address the aforementioned problems. Also, usability engineering

phases has been utilized as a methodological framework to analyze, improve, and evaluate the interactions between users and Web 2.0 sites [12]. In addition, several research approaches related to Web annotation, Web adaptation, and Web usability evaluation have been discussed from different viewpoints. These approaches has been discussed in *Chapter 2*.

Afterwards, we have analyzed several aspects related to users' interactions with Web 2.0 sites in *Part I*. Particularly, we have presented several Web 2.0 use cases and provided some practical examples for each use cases. We also have analyzed several characteristics for each use cases, with more focus on local context perspective. In addition, we have studied several types of $CSCs$ and context information that are used to represent and interpret each types of them. Accordingly, we have defined the following requirements in order to improve the usability of users' interactions: the representation of $CSCs$ have to be completed with their authors' local contexts and they have to be adapted according to their readers' local contexts. Also, the completion and adaptation of $CSCs$ have to consider the characteristics of all Web 2.0 use cases.

Based on the usability analysis, the design of Web annotation and Web adaptation have been described in *Part II*. In *Chapter 4*, we have evaluated several design alternatives to adapt $CSCs$ according to their readers' local contexts. We also have concluded that the annotation of $CSCs$ with their authors' local contexts at creation/update time and the adaptation of them at browsing time is the best design alternative with respect to the Web 2.0 use cases. Based on the adopted alternative, we have presented a semantic representation model. This model has utilized the notion of semantic object to enrich (i.e., annotate) $CSCs$ with suitable authors' context information. In addition, we have introduced an ontology called local context ontology in order to foster the interoperability of annotated $CSCs$ among users' applications. Finally, we have introduced an architecture in order to demonstrate how our approach work seamlessly with Web technology stacks.

In *Chapter 5*, we have evaluated several annotation alternatives and concluded that the use of the RDFa technology to annotate $CSCs$ with a minimum set of context attributes that are extracted/inferred from authors' contexts is the best annotation tradeoff. Then, we have developed an interactive annotation process that assists authors to specify their local context and annotate $CSCs$ with a suitable context information. Finally, we have described the internal structure of the annotation engine that is used to embody our annotation process.

In *Chapter 6*, we have developed an adaptation process. This process adapts semantic objects (i.e., annotated $CSCs$) from their multiple authors' local contexts to their

reader's local context based on a set of adaptation functions dedicated to each type of semantic object.

Finally, *Part III* has introduced an evaluation methodology. This methodology has detailed our recommendation on how to evaluate the actual users' interactions after carrying out the Web annotation and Web adaptation summarized above.

Hence, the main contribution of this thesis can be summarized as follows:

- We have evaluated several design alternatives in order to optimize the adaptation of $CSCs$. Also, a semantic representation model has been proposed based on the adopted design alternative. This model utilizes the semantic object notion and enriches $CSCs$ with suitable context information. In addition, we have introduced local context ontology in order to represent the local context information at conceptual level.

- We have also introduced an architecture in order to describe the main components that are necessary to accomplish the Web annotation and Web adaptation.

- We have evaluated several annotation alternative and introduced an interactive Web annotation process. This process details how Web authors are assisted to specify their contexts and to annotate $CSCs$ with a suitable context information.

- We have introduced Web adaptation process and a set of adaptation functions for each type of semantic objects. This process adapts semantic according to their readers' contexts at browsing time.

- Finally, we have proposed a usability evaluation methodology. This methodology explains how to evaluate users' interactions after carrying out the proposed Web annotation and Web adaptation.

8.2 Conclusions

During the last decade, the Web has evolved to a new era characterized by authoring and sharing of Web contents via different Web users and sites, known as Web 2.0. Indeed, Web users can now contribute in creating and updating Web contents, in addition to browsing them. Also, Web contents from different Web sites can be aggregated, remixed, and displayed together in a single Web page. On the other hand, Web users are increasing and they are originated from different local communities. These users implicitly follow their local contexts when they represent and interpret Web contents.

One of the challenges is to allow these users to interact with Web 2.0 according to their local contexts.

Web adaptation is one of the well-known techniques that is utilized to adapt Web contents according to their users' needs (i.e., local contexts in this work). However, the emergence of Web 2.0 raises new challenges on Web contents' adaptation. The main challenge lies into the heterogenous nature of Web contents. Indeed, different Web contents from different sources could refer to the same real-world concept. Moreover, these contents could be represented in different ways, as they are implicity represented according to their authors' local contexts. One possible way to address this challenge is to rely on Web annotation. the latter enriches Web contents with semantic matadata. However, it is extremely new technology and relying on users to manually annotate Web contents is considered difficult and prone to serious errors. This thesis proposes a combination of interactive Web annotation and Web adaptation approach to handle the local contexts of Web users, as summarized above.

8.3 Future Works

We believe that the future research in this area should be *empirical*. Currently, we have illustrated the problems that arise when Web users follow their local contexts to interact with the Web, and we have proposed a design solution for addressing them. It is now time to set up an empirical experiment in order to validate the feasibility of our approach in the real world. To do so, we have to investigate the integration/extension of our evaluation methodology with other usability evaluation methodologies utilized to evaluate other usability aspects such as the evaluation of users' interactions during Web contents' creation, update, and browsing.

Bibliography

[1] Rusli Abdullah and Koh Tieng Wei. Usability measurement of malaysia online news websites. *IJCSNS International Journal of Computer Science and Network Security*, 8(5), 2008.

[2] Gregory D. Abowd, Christopher G. Atkeson, Jason I. Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: A mobile context-aware tour guide. *Wireless Networks*, 3(5):421–433, 1997.

[3] Ben Adida. hGRDDL: Bridging microformats and RDFa. *J. Web Sem.*, 6(1), 2008.

[4] Mohanad Al-Jabari, Michael Mrissa, and Philippe Thiran. Handling users local contexts in web 2.0: Use cases and challenges. In *AP WEB 2.0 International Workshop*, pages 11–20, 2009.

[5] Mohanad Al-Jabari, Michael Mrissa, and Philippe Thiran. Towards web usability: Providing web contents according to the readers contexts. In *UMAP*, volume 5535 of *Lecture Notes in Computer Science*, pages 467–473. Springer, 2009.

[6] Mohanad Al-Jabari, Michael Mrissa, and Philippe Thiran. Context-aware interactive approach to handle users local contexts in web 2.0. In *ICWE*, 2010.

[7] Hend S. Al-Khalifa and Jessica Rubart. Automatic document-level semantic metadata annotation using folksonomies and domain ontologies. *SIGWEB Newsl.*, 2008(Autumn):1–3, 2008.

[8] Paul Anderson. What is web 2.0? ideas, technologies and implications for education. Technical report, JISC Technology and Standards Watch, 2007.

[9] Antonella De Angeli and Leantros Kyriakoullis. Globalisation vs. localisation in e-commerce: cultural-aware interaction design. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 250–253, New York, NY, USA, 2006. ACM.

[10] Anupriya Ankolekar, Markus Krötzsch, Thanh Tran, and Denny Vrandecic. The two cultures: Mashing up web 2.0 and the semantic web. *J. Web Sem.*, 6(1):70–75, 2008.

[11] Anupriya Ankolekar and Denny Vrandecic. Personalizing web surfing with semantically enriched peronal profiles. In Makram Bouzid and Nicola Henze, editors, *Proceedings of the Semantic Web Personalization Workshop*, Budva, Montenegro, JUN 2006.

[12] Nuray Aykin. *Usability and Internationalization of Information Technology (Volume in the Human Factors/Ergonomics Series)*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2004.

[13] Faical Azouaou, Weiqin Chen, and Cyrille Desmoulins. Semantic annotation tools for learning material. a part of EU IST Technology Enhanced Learning (TEL) project 507838.

[14] Faiçal Azouaou and Cyrille Desmoulins. Semantic annotation for the teacher: models for a computerized memory tool. In *Proceedings of the Third International Workshop on Applications of Semantic Web Technologies for E-Learning (SW-EL 2005)*, page 10 pages, Amsterdam Netherlands, 2005.

[15] Sandrine Balbo, Steve Goschnick, Derek Tong, and Ccile Paris. Leading web usability evaluations to wauter. In *In Proceedings of the 11th Australian World Wide Web Conference (AusWeb)*, 2005.

[16] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *IJAHUC*, 2(4):263–277, 2007.

[17] W. Barber and A. Badre. Culturability: The merging of culture and usability. In *the 4th Conference on Human Factors and the Web*, 1998.

[18] Sean Bechhofer, Simon Harper, and Darren Lunn. Sadie: Semantic annotation for accessibility. In Cruz et al. [38], pages 101–115.

[19] Abdo Beirekdar, Jean Vanderdonckt, Jean V, and Monique Noirhomme-fraiture. Kwaresmi - knowledge-based web automated evaluation tool with reconfigurable guidelines optimization, 2003.

[20] Rudi Belotti, Corsin Decurtins, Michael Grossniklaus, Moira C. Norrie, and Alexios Palinginis. Interplay of content and context. In Koch et al. [72], pages 187–200.

[21] Raquel Benbunan-Fich. Using protocol analysis to evaluate the usability of a commercial web site. *Inf. Manage.*, 39(2):151–163, 2001.

[22] V. Richard Benjamins, Dieter Fensel, Stefan Decker, and Asunción Gómez-Pérez. (ka)^2: building ontologies for the internet: a mid-term report. *Int. J. Hum.-Comput. Stud.*, 51(3):687–712, 1999.

[23] Hannes Bohring and Sren Auer. Mapping xml to owl ontologies. In *Leipziger Informatik-Tage, volume 72 of LNI*, pages 147–156. GI, 2005.

[24] Christof Bornhövd. Semantic metadata for the integration of web-based data for electronic commerce. In *WECWIS '99: Proceedings of the International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems*, page 137, Washington, DC, USA, 1999. IEEE Computer Society.

[25] Danah Boyd and Nicole B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1-2), November 2007.

[26] Peter Brusilovsky. Adaptive hypermedia. *User Model. User-Adapt. Interact.*, 11(1-2):87–110, 2001.

[27] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors. *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*. Springer, 2007.

[28] Michel Buffa, Fabien L. Gandon, Guillaume Erétéo, Peter Sander, and Catherine Faron. Sweetwiki: A semantic wiki. *J. Web Sem.*, 6(1):84–97, 2008.

[29] Juan Cappi, Gustavo Rossi, Andres Fortier, and Daniel Schwabe. Seamless personalization of e-commerce applications. In *Revised Papers from the HUMACS, DASWIS, ECOMO, and DAMA on ER 2001 Workshops*, pages 457–470, London, UK, 2002. Springer-Verlag.

[30] Les Carr, Timothy Miles-Board, Arouna Woukeu, Gary Wills, and Wendy Hall. The case for explicit knowledge in documents. In Ethan V. Munson and Jean-Yves Vion-Dury, editors, *ACM Symposium on Document Engineering*, pages 90–98. ACM, 2004.

[31] Stefano Ceri, Florian Daniel, Maristella Matera, and Federico Michele Facca. Model-driven development of context-aware web applications. *ACM Trans. Internet Techn.*, 7(1), 2007.

[32] Stefano Ceri, Peter Dolog, Maristella Matera, and Wolfgang Nejdl. Model-driven design of web applications with client-side adaptation. In Koch et al. [72], pages 201–214.

[33] Po-Hao Chang and Gul Agha. Towards context-aware web applications. In Jadwiga Indulska and Kerry Raymond, editors, *DAIS*, volume 4531 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 2007.

[34] Philipp Cimiano, Günter Ladwig, and Steffen Staab. Gimme' the context: context-driven automatic semantic annotation with c-pankow. In Allan Ellis and Tatsuya Hagino, editors, *WWW*, pages 332–341. ACM, 2005.

[35] Michael Cooper. Evaluating accessibility and usability of web pages. In *Proceedings of the third international conference on Computer-aided design of user interfaces*, pages 33–42, Norwell, MA, USA, 1999. Kluwer Academic Publishers.

[36] Óscar Corcho. Ontology based document annotation: trends and open research problems. *IJMSO*, 1(1):47–57, 2006.

[37] Graham Cormode and Balachander Krishnamurthy. Key differences between web 1.0 and web 2.0. *First Monday*, 13(6), 2008.

[38] Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors. *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*. Springer, 2006.

[39] Dianne Cyr and Haizley Trevor-Smith. Localization of web design: An empirical comparison of german, japanese, and united states web site characteristics. *JASIST*, 55(13):1199–1208, 2004.

[40] Stefan Decker, Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The semantic web: The roles of xml and rdf. *IEEE Internet Computing*, 4(5):63–74, 2000.

[41] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[42] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 178–186, New York, NY, USA, 2003. ACM.

[43] Brahim Djioua, Jorge J. García Flores, Antoine Blais, Jean-Pierre Desclés, Gaëll Guibert, Agata Jackiewicz, Florence Le Priol, Leila Nait-Baha, and Benoît Sauzay. Excom: An automatic annotation engine for semantic information. In Geoff Sutcliffe and Randy Goebel, editors, *FLAIRS Conference*, pages 285–290. AAAI Press, 2006.

[44] Jérôme Euzenat. Eight questions about semantic web annotations. *IEEE Intelligent Systems*, 17(2):55–62, 2002.

[45] Piero Fraternali, Maristella Matera, and Andrea Maurino. Conceptual-level log analysis for the evaluation of web application quality. In *LA-WEB*, pages 46–57. IEEE Computer Society, 2003.

[46] Cristina Gena and Stephan Weibelzahl. Usability engineering for the adaptive web. In Brusilovsky et al. [27], pages 720–762.

[47] Anna Goy, Liliana Ardissono, and Giovanna Petrone. Personalization in e-commerce applications. In Brusilovsky et al. [27], pages 485–520.

[48] Alexander Graf. Rdfa vs. microformats. Technical report, DERI, 04 2007.

[49] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, 1995.

[50] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5-6):907 – 928, 1995.

[51] Ramanathan V. Guha and Rob McCool. Tap: a semantic web platform. *Computer Networks*, 42(5):557–577, 2003.

[52] Siegfried Handschuh and Steffen Staab. Authoring and annotation of web pages in cream. In *WWW*, pages 462–473, 2002.

[53] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-cream - semi-automatic creation of metadata. In Asunción Gómez-Pérez and V. Richard Benjamins, editors, *EKAW*, volume 2473 of *Lecture Notes in Computer Science*, pages 358–372. Springer, 2002.

[54] Ilse Maria Harms and Werner Schweibenz. Usability engineering methods for the web: Results from a usability study. In Gerhard Knorz and Rainer Kuhlen, editors, *ISI*, volume 38 of *Schriften zur Informationswissenschaft*, pages 17–30. Hochschulverband für Informationswissenschaft, 2000.

[55] Simon Harper and Sean Bechhofer. Semantic triage for increased web accessibility. *IBM Systems Journal*, 44(3):637–649, 2005.

[56] Jeff Heflin, James Hendler, and Sean Luke. Reading between the lines: Using shoe to discover implicit knowledge from the web. In *In AI and Information Integration, Papers from the 1998 Workshop, Menlo Park, CA*. AAAI Press, 1998.

[57] Banati Hema, Bedi Punam, and Grover P.S. Evaluating web usability from the user's perspective. *Journal of Computer Science*, 2(4):314–317, 2006.

[58] Masahiro Hori, Mari Abe, and Kouichi Ono. Extensible framework of authoring tools for web document annotation. In *International Workshop on Semantic Web Foundations and Application Technologies (SWFAT)*, Japan Nara, 2003.

[59] Masahiro Hori, Goh Kondoh, Kouichi Ono, Shin'ichi Hirose, and Sandeep K. Singhal. Annotation-based web content transcoding. *Computer Networks*, 33(1-6):197–211, 2000.

[60] David Huynh, Stefano Mazzocchi, and David R. Karger. Piggy bank: Experience the semantic web inside your web browser. *J. Web Sem.*, 5(1):16–27, 2007.

[61] Eero Hyvönen and Eetu Mäkelä. Semantic autocompletion. In Riichiro Mizoguchi, Zhongzhi Shi, and Fausto Giunchiglia, editors, *ASWC*, volume 4185 of *Lecture Notes in Computer Science*, pages 739–751. Springer, 2006.

[62] Peter Sandrini (Innsbruck). Website localization and translation. In Heidrun Gerzymisch-Arbogast (Saarbrcken)and Sandra Nauert (Saarbrcken), editor, *MuTra 2005 - EU-High-Level Scientific Conference:Challenges of Multidimensional Translation*, May 2005.

[63] Melody Y. Ivory and Marti A Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv.*, 33(4):470–516, 2001.

[64] Tatjana Jevsikova. Localization and internationalization of web-based learning environment. In Roland Mittermeir, editor, *ISSEP*, volume 4226 of *Lecture Notes in Computer Science*, pages 310–318. Springer, 2006.

[65] José Kahan and Marja-Riitta Koivunen. Annotea: an open rdf infrastructure for shared web annotations. In *WWW*, pages 623–632, 2001.

[66] José Kahan, Marja-Riitta Koivunen, Eric Prud' hommeaux, and Ralph R. Swick. Annotea: an open rdf infrastructure for shared web annotations. *Computer Networks*, 39(5), 2002.

[67] Joachim Wolfgang Kaltz. *An Engineering Method for Adaptive, Context-aware Web Applications*. Phd thesis, Fakultt fr Ingenieurwissenschaften " Ingenieurwissenschaften - Campus Duisburg " Abteilung Informatik und Angewandte Kognitionswissenschaft, 2006.

[68] Gerti Kappel, Birgit Pröll, Werner Retschitzegger, and Wieland Schwinger. Modelling ubiquitous web applications - the wuml approach. In Hiroshi Arisawa, Yahiko Kambayashi, Vijay Kumar, Heinrich C. Mayr, and Ingrid Hunt, editors, *ER (Workshops)*, volume 2465 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2001.

[69] Brian P. Kettler, James Starz, William Miller, and Peter Haglich. A template-based markup tool for semantic web content. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 446–460. Springer, 2005.

[70] Rohit Khare. Microformats: The next (small) thing on the semantic web? *IEEE Internet Computing*, 10(1):68–75, 2006.

[71] Alfred Kobsa, J. Koenemann, and W. Pohl. Personalized hypermedia presentation techniques for improving online customer relationships. *The Knowledge Engineering Review*, 16(2):111–155, 2001.

[72] Nora Koch, Piero Fraternali, and Martin Wirsing, editors. *Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004, Proceedings*, volume 3140 of *Lecture Notes in Computer Science*. Springer, 2004.

[73] Ron Kohavi, Randal M. Henne, and Dan Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 959–967, New York, NY, USA, 2007. ACM.

[74] Markus Krötzsch, Denny Vrandecic, and Max Völkel. Semantic mediawiki. In Cruz et al. [38], pages 935–942.

[75] Tayeb Lemlouma and Nabil Layada. Smil content adaptation for embedded devices. In *in SMIL Europe 2003 Conference*, pages 12–14, 1214.

[76] James R. Lewis. Ibm computer usability satisfaction questionnaires:psychometric evaluation and instructions for use. Technical Report 54.786, IBM Corporation, Human Factors Group, 1993.

[77] Ralf Heese Markus Luczak-Roesch. Linked data authoring for non-experts. In *Proceedings of the WWW09, Workshop Linked Data on the Web (LDOW2009)*, 2009.

[78] P. Markellou, I. Mousourouli, S. Spiros, and A. Tsakalidis. Using semantic web mining technologies for personalized e-learning experiences. In *web-based education (pp. 461-826).*, 2005.

[79] Maristella Matera, Francesca Rizzo, and Giovanni Toffetti Carughi. Web usability: Principles and evaluation methods. In *Web Engineering*, pages 143–180. Springer, 2006.

[80] Sean M. McNee, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Interfaces for eliciting new user preferences in recommender systems. In Peter Brusilovsky, Albert T. Corbett, and Fiorella de Rosis, editors, *User Modeling*, volume 2702 of *Lecture Notes in Computer Science*, pages 178–187. Springer, 2003.

[81] Michael Mrissa, Mohanad Al-Jabari, and Phillippe Thiran. Using microformats to personalize web experience. In *Proceedings of the 7th International Workshop on Web-Oriented Software Technologies (IWWOST08)*, 2008.

[82] Michael Mrissa, Chirine Ghedira, Djamal Benslimane, and Zakaria Maamar. A context model for semantic mediation in web services composition. In *ER*, pages 12–25, 2006.

[83] Christine Muller and Michael Kohlhase. Context-aware adaptation: A case study on mathematical notations. *Inf. Sys. Manag.*, 26(3):215–230, 2009.

[84] Deirdre Mulligan and Ari Schwartz. Your place or mine?: privacy concerns and solutions for server and client-side storage of personal information. In *CFP '00: Proceedings of the tenth conference on Computers, freedom and privacy*, pages 81–84, New York, NY, USA, 2000. ACM.

[85] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[86] Jakob Nielsen. Web 2.0 can be dangerous... http://www.useit.com/alertbox/web-2.html, December 2007.

[87] Donald A. Norman. *The Design of Everyday Things*. Basic Books, September 2002.

[88] Deborah L. McGuinness Ora Lassila. The role of frame-based representation on the semantic web. *Electronic Transactions on Artificial Intelligence*, 6(005), 2001.

[89] Tim O'Reilly. What is web 2.0? design patterns and business models for the next generation of software. 2005.

[90] Frederik Pfisterer, Markus Nitsche, Anthony Jameson, and Catalin Barbu. User-centered design and evaluation of interface enhancements to the semantic mediawiki. In *Exploring HCI Challenges*, volume 541, Italy, April 2008. CEUR-WS.

[91] Valentina Presutti and Aldo Gangemi. Content ontology design patterns as practical building blocks for web ontologies. In *ER '08: Proceedings of the 27th International Conference on Conceptual Modeling*, pages 128–141, Berlin, Heidelberg, 2008. Springer-Verlag.

[92] Lawrence H. Reeve and Hyoil Han. Survey of semantic annotation platforms. In Hisham Haddad, Lorie M. Liebrock, Andrea Omicini, and Roger L. Wainwright, editors, *SAC*, pages 1634–1638. ACM, 2005.

[93] Sara Rigutti and Gisella Paoletti. Web 2.0: which usability issues? *E-Learning and Knowledge Society (Je-LKS)*, 4(2):229 – 234, June 2008.

[94] Gustavo Rossi, Daniel Schwabe, and aes Robson Guimar˙ Designing personalized web applications. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 275–284, New York, NY, USA, 2001. ACM.

[95] Segawa Satoko, Sugimura Masahiko, and Ishigaki Kazushi. New web-usability evaluation method: scenario-based walkthrough. *FUJITSU Scientific and Technical Journal*, 41:1:105–114, 2005.

[96] Subramanian Sattanathan, Nanjangud C. Narendra, and Zakaria Maamar. Towards context-based tracking of web services security. In Gabriele Kotsis, David Taniar, Stéphane Bressan, Ismail Khalil Ibrahim, and Salimah Mokhtar, editors, *iiWAS*, volume 196 of *books@ocg.at*, pages 13–24. Austrian Computer Society, 2005.

[97] Albrecht Schmidt, Michael Beigl, and Hans-Werner Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

[98] Kay-Uwe Schmidt, Jörg Dörflinger, Tirdad Rahmani, Mehdi Sahbi, Ljiljana Stojanovic, and Susan Marie Thomas. An user interface adaptation architecture for rich internet applications. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *ESWC*, volume 5021 of *Lecture Notes in Computer Science*, pages 736–750. Springer, 2008.

[99] Daniel Schwabe, Gustavo Rossi, and Simone Diniz Junqueira Barbosa. Systematic hypermedia application design with oohdm. In *Hypertext*, pages 116–128. ACM, 1996.

[100] Edward Sciore, Michael Siegel, and Arnon Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Trans. Database Syst.*, 19(2):254–290, 1994.

[101] Brian Shackel. Usability—context, framework, definition, design and evaluation. pages 21–37, 1991.

[102] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[103] Reetta Sinkkila, Eetu Makela, Tomi Kauppinen, and Eero Hyvonen. Combining context navigation with semantic autocompletion to solve problems in concept selection. In Khalid Belhajjame, Mathieu d'Aquin, Peter Haase, and Paolo Missier, editors, *First International Workshop on Semantic Metadata Management and Applications, SeMMA 2008, Located at the Fifth European Semantic Web Conference (ESWC 2008), Tenerife, Spain, June 2nd, 2008. Proceedings*, volume 346 of *CEUR Workshop Proceedings*, pages 61–68. CEUR-WS.org, June 1-5 2008.

[104] Markus Specker and Ina Wentzlaff. Exploring usability needs by human-computer interaction patterns. In Marco Winckler, Hilary Johnson, and Philippe A. Palanque, editors, *TAMODIA*, volume 4849 of *Lecture Notes in Computer Science*, pages 254–260. Springer, 2007.

[105] Hlne Stengers, Olga De Troyer, Martine Baetens, Frank Boers, and Abdalghani N.Mushtaha. Localization of web sites: Is there still a need for it? In International Workshop on Web Engineering (held in conjunction with the ACM HyperText 2004 Conference), 2004.

[106] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey, September 2004.

[107] Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. Cool: A context ontology language to enable contextual interoperability. In Jean-Bernard Stefani, Isabelle M. Demeure, and Daniel Hagimont, editors, *DAIS*, volume 2893 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 2003.

[108] Elias Torres. Open data in xhtml, 2007. XTECH CONFERENCE 2007.

[109] Olga De Troyer and Sven Casteleyn. Designing localized web sites. In Xiaofang Zhou, Stanley Y. W. Su, Mike P. Papazoglou, Maria E. Orlowska, and Keith G. Jeffery, editors, *WISE*, volume 3306 of *Lecture Notes in Computer Science*, pages 547–558. Springer, 2004.

[110] Tom Tullis, Stan Fleischman, Michelle McNulty, Carrie Cianchette, and Marguerite Bergel. An empirical comparison of lab and remote usability testing of web sites. In *Usability Professionals Conference (UPA)*, Pennsylvania, 2002.

[111] Victoria S. Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *J. Web Sem.*, 4(1):14–28, 2006.

[112] Onni Valkeapää, Olli Alm, and Eero Hyvönen. An adaptable framework for ontology-based content creation on the semantic web. *J. UCS*, 13(12):1835–1835, 2007.

[113] Roberto De Virgilio and Riccardo Torlone. Management of heterogeneous profiles in context-aware adaptive information system. In Robert Meersman, Zahir Tari,

Pilar Herrero, Gonzalo Méndez, Lawrence Cavedon, David Martin, Annika Hinze, George Buchanan, María S. Pérez, Víctor Robles, Jan Humble, Antonia Albani, Jan L. G. Dietz, Hervé Panetto, Monica Scannapieco, Terry A. Halpin, Peter Spyns, Johannes Maria Zaha, Esteban Zimányi, Emmanuel Stefanakis, Tharam S. Dillon, Ling Feng, Mustafa Jarrar, Jos Lehmann, Aldo de Moor, Erik Duval, and Lora Aroyo, editors, *OTM Workshops*, volume 3762 of *Lecture Notes in Computer Science*, pages 132–141. Springer, 2005.

[114] Martijn Van Welie, Gerrit C. Van Der Veer, and Anton Elins. Breaking down usability. In *Proceedings of Interact '99*, pages 613–620. Press, 1999.

[115] Yeliz Yesilada, Simon Harper, Carole Goble, and Robert Stevens. Ontology based semantic annotation for enhancing mobility support for visually impaired web users. In *In K-CAP 2003 Workshop on Knowledge Markup and Semantic Annotation*, 2003.

[116] Yan Zhu, Christof Bornhövd, Doris Sautner, and Alejandro P. Buchmann. Materializing web data for olap and dss. In *WAIM '00: Proceedings of the First International Conference on Web-Age Information Management*, pages 201–214, London, UK, 2000. Springer-Verlag.

[117] Alexander Zipf and Matthias Jöst. Implementing adaptive mobile gi services based on ontologies: Examples from pedestrian navigation support. *Computers, Environment and Urban Systems*, 30(6):784–798, 2006.